Just Fast Keying in the Pi Calculus

Martín Abadi
University of California, Santa Cruz, and Microsoft Research
and
Bruno Blanchet
CNRS, École Normale Supérieure, Paris
and
Cédric Fournet
Microsoft Research

JFK is a recent, attractive protocol for fast key establishment as part of securing IP communication. In this paper, we analyze it formally in the applied pi calculus (partly in terms of observational equivalences, partly with the assistance of an automatic protocol verifier). We treat JFK's core security properties, and also other properties that are rarely articulated and studied rigorously, such as plausible deniability and resistance to denial-of-service attacks. In the course of this analysis we found some ambiguities and minor problems, such as limitations in identity protection, but we mostly obtain positive results about JFK. For this purpose, we develop ideas and techniques that should be useful more generally in the specification and verification of security protocols.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—security and protection; C.2.2 [**Computer-Communication Networks**]: Network protocols—protocol verification; D.2.4 [**Software Engineering**]: Software/Program Verification—correctness proofs, formal methods; F.3.1 [**Theory of Computation**]: Specifying and Verifying and Reasoning about Programs—mechanical verification, specification techniques

General Terms: Languages, Security, Theory, Verification

Additional Key Words and Phrases: IP security, Key exchange, Process calculus

1. INTRODUCTION

The design of security mechanisms for the Internet has been the focus of much activity. In particular, IP security has received much attention; in this area, we have seen some progress but also some disappointment and some controversy. The Internet Key Exchange (IKE) protocol [Harkins and Carrel 1998], an important method for establishing cryptographic keys for secure IP communication, has been the subject of considerable and reasonable criticisms. Those criticisms tend to concern not the core authenticity and secrecy properties

Authors' addresses: M. Abadi, abadi@cs.ucsc.edu; B. Blanchet, blanchet@di.ens.fr; C. Fournet, fournet@microsoft.com.
An earlier version of this paper was presented at the 13th European Symposium on Programming (ESOP 2004).

that IKE offers but rather the complexity of IKE, some of its inefficiencies, and its poor resistance against denial-of-service (DOS) attacks. Several recent protocols aim to address IKE's shortcomings. These include the JFK protocol [Aiello et al. 2002b; 2002a] (for "just fast keying") and the IKEv2 protocol [Kaufman 2005].

In some respects, IKE and its successors are fairly classical security protocols. They all employ common pieces in the standard arsenal of modern cryptography, and aim to guarantee the integrity and secrecy of IP communication. They are all subject to common efficiency considerations, which limit the use of expensive cryptographic operations and the number and size of messages. Beyond such basic aspects, however, these protocols—and JFK in particular—exhibit a number of interesting features because they address other security objectives. These other objectives are sometimes subtle; they are seldom articulated precisely. Moreover, they give rise to new tensions and delicate compromises. For instance, in the name of privacy, a protocol may attempt to hide the identities of the participants (that is, to provide identity protection) and to guarantee the plausible deniability of their actions, and may accordingly avoid or delay the authentication of the participants. On the other hand, strong, early authentication can simplify DOS resistance. Of course, such tensions are not unique to JFK and its close relatives. Rather, they seem to be increasingly important in the design of modern security protocols. JFK exemplifies them well and resolves them nicely.

In this paper we analyze JFK, relying on the applied pi calculus, an extension of the standard pi calculus with functions. Specifically, we present a formalization of JFK in the applied pi calculus; we focus on a variant of JFK known as JFKr (which is the closer one to IKEv2), but we also consider its other major variant, JFKi, more briefly. While fairly short and abstract, our formalization gives a fine level of detail in the modelling of contexts and parallel sessions. It also covers aspects of the protocol beyond the "messages on the wire", such as protocol interfaces, the checks performed by the participants, and other delicate features such as the treatment of duplicate requests.

We treat the core security properties of the protocol, and also other properties that are rarely articulated and studied rigorously, such as plausible deniability and DOS resistance. (We consider all the properties with a single model of the protocol: we do not need to define special, partial models for particular properties.) We also provide proofs for those properties. Some of the proofs were done by hand, while others were done with an automated protocol verifier, ProVerif [Blanchet 2001]. In some cases, there are overlaps between the two kinds of proofs; those overlaps provide extra assurance about the correctness of the formalization and the proofs. Moreover, while ProVerif can be used for establishing standard security properties such as correspondence assertions, it is still limited when it comes to subtler properties, which we therefore prove partly by hand.

In the course of this analysis, we identified some minor limitations and weaknesses of JFK. In particular, we discovered that JFK does not provide as much identity protection as one might have expected on the basis of informal descriptions of the protocol. However, we did not discover fatal mistakes. That is comforting but not surprising, since the authors of JFK have substantial experience in protocol design and since JFK benefited from careful review and prolonged discussion in the IETF context.

Beyond observations and results on JFK, this study contributes to the specification and verification of security protocols in several ways. Our basic approach and tools come from recent work; it is pleasing to confirm their effectiveness. On the other hand, the approach to formalizing several of the protocol's less mundane facets is largely new, and should be applicable elsewhere. Similarly, the proofs are non-trivial and motivate some new developments of our techniques. These novelties include a formulation of plausible deniability, a general lemma about state elimination, and extensions in ProVerif. The proofs also provide an opportunity for integrating manual and automatic methods in the applied pi calculus. This integration relies on new results on the correctness of ProVerif proofs.

Contents. The next section is a review and informal discussion of JFK. Section 3 introduces the applied pi calculus. Section 4 then presents a model of JFKr in the applied pi calculus. Section 5 discusses ProVerif, its extensions, and its use. Section 6 treats DOS resistance. Section 7 concerns core security properties (secrecy and authenticity). It also briefly addresses identity protection. Section 8 deals with plausible deniability. Section 9 mentions some related work and concludes. The appendix includes details on our use of ProVerif and proofs. Our ProVerif scripts are on-line, at `http://www.di.ens.fr/~blanchet/crypto/jfk.html`.

2. THE JFK PROTOCOL

The JFK protocol has been discussed in a series of five Internet Drafts [Aiello et al. 2002b], starting in 2001, and it is also described in a conference paper [Aiello et al. 2002a] and in a journal paper [2004]. While our work is based on all those documents, we tend to privilege the contents of the papers, since they should have more permanence than the Internet Drafts. Primarily we follow the presentation of the conference paper; we also discuss minor changes introduced in the journal paper. We refer to the papers for additional material on the protocol and its motivation.

JFK involves two principals that play the roles of an initiator (I) and a responder (R). As in many other protocols, these two principals wish to open a secure communication channel, and they attempt to accomplish it by establishing a shared secret. This shared secret serves as the basis for computing session keys. The two principals should associate the shared secret with each other, verify each other's identities, and also agree on various communication parameters (for example, what sort of session keys to employ). Attackers may eavesdrop, delete, and insert messages; they may also attempt to impersonate principals [Needham and Schroeder 1978]. Therefore, the communications between the initiator and the responder are cryptographically protected.

JFK has two major variants, JFKr and JFKi. These differ in their protection of identity information. JFKr aims to protect the identity of the responder against active attacks, and also the identity of the initiator against passive attacks. JFKi aims to protect the identity of the initiator against active attacks.

2.1 The JFKr Variant

The JFKr protocol consists of the following four messages:

Message 1	$I \rightarrow R:$	N_I, x_I
Message 2	$R \rightarrow I:$	$N_I, N_R, x_R, \mathrm{g}_R, t_R$
Message 3	$I \rightarrow R:$	$N_I, N_R, x_I, x_R, t_R, e_I, h_I$
Message 4	$R \rightarrow I:$	e_R, h_R

$z = I, R$	one of the two roles in the protocol: initiator or responder.
i, r	two distinct constants used to tag initiator and responder MACs.
N_z	random fresh nonce for the session.
d_z	Diffie-Hellman secret exponents.
$x_z = \mathsf{g}\hat{\ }d_z$	Diffie-Hellman exchange values (g^i and g^r in [Aiello et al. 2002a]).
g	Diffie-Hellman group (possibly obtained from a previously received g_R).
g_R	responder's choice of group g and algorithms (*GRPINFO*$_R$ in [Aiello et al. 2002a]).
t_R	authenticator cookie used by the responder against DOS.
K_R	responder's secret hash key for authenticators t_R (*HK*$_R$ in [Aiello et al. 2002a]).
$u = a, e, v$	one of the three usages for keys: authentication, encryption, and main session secret.
a, e, v	three distinct constants used to tag usages for keys.
K_u	shared key obtained by a Diffie-Hellman computation, specialized for u.
E	shared-key encryption function.
H	keyed hash function for MACs (message authentication codes).
e_z, h_z	encrypted payload messages and their MACs (protecting z's identity and signature).
S	public-key signature function.
s_z	signed nonces and exponentials.
K_-^z	private signature key for the principal playing role z.
ID_z	identity for the principal playing role z, and its public signature-verification key.
ID'_R	"hint" of the responder identity, provided by the initiator.
IP_I	IP source address for the initiator (hashed in t_R).
sa_z	additional parameters for IP security associations (sa and sa' in [Aiello et al. 2002a]).
A, B	principals taking part in the protocol (in either or both roles).

Fig. 1. Main notations

where:

$$
\begin{array}{rclrcl}
x_I & = & \mathsf{g}\hat{\ }d_I & x_R & = & \mathsf{g}\hat{\ }d_R \\
 & & & t_R & = & \mathsf{H}\{K_R\}(x_R, N_R, N_I, \mathsf{IP}_I) \\
K_u & = & \mathsf{H}\{x_R\hat{\ }d_I\}(N_I, N_R, \mathsf{u}) \quad \text{for } u = a, e, v & & & \\
e_I & = & \mathsf{E}\{K_e\}(\mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, s_I) & e_R & = & \mathsf{E}\{K_e\}(\mathsf{ID}_R, \mathsf{sa}_R, s_R) \\
h_I & = & \mathsf{H}\{K_a\}(\mathsf{i}, e_I) & h_R & = & \mathsf{H}\{K_a\}(\mathsf{r}, e_R) \\
s_I & = & \mathsf{S}\{K_-^I\}(N_I, N_R, x_I, x_R, \mathsf{g}_R) & s_R & = & \mathsf{S}\{K_-^R\}(x_R, N_R, x_I, N_I)
\end{array}
$$

Figure 1 summarizes the notations of this exchange, adapted from Aiello et al. [2002b]. In particular, keyed cryptographic primitives take a key as first argument in braces, followed by other arguments in parentheses—for instance $\mathsf{E}\{K\}(T)$ is the encryption of plaintext T under key K.

The first pair of messages establishes a shared secret via a Diffie-Hellman exchange. Each principal generates and communicates a fresh nonce N_z. Each principal also selects or generates a secret exponent d_z, and communicates the corresponding exponential $x_z = \mathsf{g}\hat{\ }d_z$. Relying on the equation $x_R\hat{\ }d_I = x_I\hat{\ }d_R$, three independent shared keys are derived from nonces and exponentials: K_a and K_e are used in Messages 3 and 4, while K_v is returned to each principal as the newly established session secret. The reuse of exponentials is allowed, with a trade-off between forward secrecy and efficiency; in any case, the freshness of nonces suffices to guarantee that the generated shared secrets differ for all sessions.

Message 2 includes an authenticator cookie t_R, keyed with a secret local to the responder, K_R. The responder expects to see this cookie in Message 3, as proof of a successful round-trip (Messages 1 and 2), and it need not perform any expensive cryptography or allocate resources before validating the cookie. Furthermore, after receiving Message 3, the responder can remember handling t_R, so as to avoid expense in the event that t_R is replayed.

The second pair of messages provides authentication. Specifically, Messages 3 and 4 include encrypted signatures of the nonces, exponentials, and other material. The encryptions protect identity information. The signatures can be interpreted as delegations from the principals that control the signature keys (possibly users) to the protocol endpoints that control the secret exponents. Only transient protocol data is signed—not identities or long-term keys associated with users. In this respect, the protocol is in tune with concerns about plausible deniability that have appeared from time to time in this context.

A recent refinement of JFK uses a hash of N_I instead of N_I in the first two messages and within signatures [Aiello et al. 2004]. This change "raises the bar" for DOS attacks in certain environments where the attacker can eavesdrop and inject messages but not modify them in flight. We have redone our automated proofs with this change, both for JFKr and JFKi; we have not revisited our manual proofs.

2.2 The JFKi Variant

JFKi uses the following messages:

$$
\begin{array}{lll}
\text{Message 1} & I \rightarrow R: & N_I, x_I, \mathsf{ID}'_R \\
\text{Message 2} & R \rightarrow I: & N_I, N_R, x_R, \mathsf{g}_R, \mathsf{ID}_R, s'_R, t_R \\
\text{Message 3} & I \rightarrow R: & N_I, N_R, x_I, x_R, t_R, e_I, h_I \\
\text{Message 4} & R \rightarrow I: & e_R, h_R
\end{array}
$$

where x_I, x_R, t_R, and K_u for $u = a, e, v$ are computed as in JFKr, but with different encrypted and signed fields:

$$
\begin{array}{rclrcl}
 & & & s'_R & = & \mathsf{S}\{K_-^R\}(x_R, \mathsf{g}_R) \\
e_I & = & \mathsf{E}\{K_e\}(\mathsf{ID}_I, \mathsf{sa}_I, s_I) & e_R & = & \mathsf{E}\{K_e\}(s_R, \mathsf{sa}_R) \\
h_I & = & \mathsf{H}\{K_a\}(\mathsf{i}, e_I) & h_R & = & \mathsf{H}\{K_a\}(\mathsf{r}, e_R) \\
s_I & = & \mathsf{S}\{K_-^I\}(N_I, N_R, x_I, x_R, \mathsf{ID}_R, \mathsf{sa}_I) & s_R & = & \mathsf{S}\{K_-^R\}(N_I, N_R, x_I, x_R, \\
 & & & & & \qquad \mathsf{ID}_I, \mathsf{sa}_I, \mathsf{sa}_R)
\end{array}
$$

The hint ID'_R appears in clear in the first message, rather than in e_I. The second message also contains a signature s'_R of the responder's parameters. The form of the last two messages is unchanged, but the parties sign each other's identity. This design choice implies that the parties cannot later deny their intent to communicate.

Although this paper presents a formal model only for JFKr, we also developed a model for JFKi and conducted all the corresponding automated proofs.

2.3 Discussion: Ambiguities and Limitations

The protocol specification, although clear, focuses on the messages exchanged in a single successful run of the protocol. It does not say much on the local processing that the parties perform, on the deployment of the protocol, and other subjects relevant for security. For instance, it does not prescribe how principals should use the protocol (and especially what is the sharing of signing keys and Diffie-Hellman exponentials); how messages should be checked; and how the responder should manage state in order to resist DOS attacks. We have reason to believe that implementations differ in some of these respects, sometimes with unfortunate consequences. The protocol specification does however state several security objectives. We discuss them all and study them formally below.

In the course of the analysis of JFK, we found some ambiguities and limitations, which we document here. These are generally minor in comparison with the positive results of the rest of the paper.

Identity Protection. Identity protection is discussed only informally in the published descriptions of JFK. Those descriptions do not provide a precise definition of identity protection, and several are possible (see Abadi and Fournet [2004]). Apparently, JFK primarily aims to conceal the identities of participants, much like one would try to hide a password or any other secret value. On the other hand, identities (for example, the name of a long-lived server) are often widely known, and JFK provides only limited protection for the fact that certain known principals communicate. We note the following leaks of information:

—A passive attacker can perform traffic analysis on JFK exchanges. For instance, if principals are associated with fixed or long-lived IP addresses, then the attacker may link their sessions.

—Similarly, if exponentials are reused at most by a single principal, then all sessions that share the same exponential must involve the same principal, and information on that principal can be correlated. In particular, when a principal A uses JFKi as both initiator and responder with a single exponential, an attacker may eavesdrop the exponential when A acts as initiator, then discover A's identity by initiating another session in which A is the responder.

—In JFKr, the identity hint ID'_R provided in Message 3 can be obtained by an active attacker that impersonates a responder, and may leak information on the intended ID_R. The attacker learns only the initiator's intent, rather than the presence of ID_R, but this presence may then be inferred, for instance by observing a successful retry.

—More remarkably, even when exponentials are not reused, ID'_R is left empty, and IP addresses are not linked to identities, an active attack against JFKr can reveal that two particular principals A and B are communicating. This attack may be hard to detect. It relies on an indirect equality test on authenticator keys, as detailed below.

(1) A initiates a session with B.
(2) C (the attacker) plays the role of responder in B's place, thus learning A's identity from Message 3 (as permitted in JFKr). C does not send any Message 4; the session fails.
(3) A retries, initiating a second session with B, with a nonce N_I.
(4) C intercepts Messages 1 and 2 of this session, then initiates yet another session with B, using the same nonce N_I but its own exponential and identity.
(5) C "swaps" these two sessions, continuing its session with the Message 2 from A's session, and forwarding to A the other Message 2.
(6) If the two sessions succeed—that is, B is willing to respond to both A and C—then C can infer that another principal has just established a session with B. In addition, C may infer that the principal is A, from the retry behavior or by other means. In short, C learns that A and B are communicating.

This attack succeeds because the hash computation of t_R keeps track of N_I but not of x_I. The omission of x_I in the computation of t_R is a deliberate performance optimization.

Fortunately, the same effect can be achieved by using $\mathsf{H}\{N_I\}(x_I)$ instead of N_I (or the hash of N_I [Aiello et al. 2004]) in Message 1, and possibly omitting x_I in that message. With this change, the scenario above becomes impossible: the two swapped sessions always fail.

Of course, any limitations of identity protection should be kept in perspective, taking into account the relative importance of various identity-protection properties.

Despite these limitations, we do get some strong, formal identity-protection guarantees, as corollaries of the theorems of Section 7. We refer to previous work on another protocol [Abadi and Fournet 2004] for a more thorough study of identity protection that relies on observational equivalences between protocol configurations with different identities. We have applied the approach developed there to JFK, finding the issues discussed above. In particular, we discovered the attack based on indirect equality tests of authenticator keys when we attempted to prove such an observational equivalence by bisimulation.

A Brief Negotiation. Protocols such as IKE and SSL include the capability of negotiating options, including cryptographic algorithms and their parameters. Some options may be weaker than others, for a variety of legal and technical reasons; principals may prefer strong options but be prepared to use weaker ones if their interlocutors require it. Since preferences can be falsified before they are authenticated, negotiation can be a source of concerns and subtle vulnerabilities (see e.g. Wagner and Schneier [1996]). In reaction, the presentations of JFK emphasize the absence of negotiation. Nevertheless, the choice of a particular Diffie-Hellman group and cryptographic algorithms by the responder can be construed as a minimal negotiation (called a "ukase" by Aiello et al. [2004]).

In JFKi, the choice is signed in Message 2; this signature is a relatively recent precaution not present in early drafts of the protocol. An analogous precaution has been omitted from JFKr because it would break identity protection. Therefore, an attacker may tamper with Message 2 in JFKr, and then either the initiator rejects the attacker's choice, interrupting communications, or the initiator sends poorly protected identity information in Message 3. (After the fact, the tampering is detected when the responder fails to verify s_I in Message 3.) In short, the tampering appears as a minor risk; hence we have decided not to model negotiation below.

Caching Answers to Message 3. The responder caches answers to Message 3, so as to answer only once for every valid authenticator cookie received in an instance of Message 3. The descriptions of JFK are somewhat ambiguous on this point. Where they refer to a duplicate Message 3, we should probably read a Message 3 with a duplicate cookie, for otherwise several problems appear. In particular, a blind DOS attack may effectively reuse a single valid cookie in numerous, cheaply generated instances of Message 3. Moreover, a responder that processes several different instances of Message 3 with the same cookie (but for example for different initiator identities) could end up with the same key for several sessions; confusion may result.

3. AN APPLIED PI CALCULUS FOR JFK

In this section, we present the instance of the applied pi calculus that we use for modelling JFKr. This calculus is an extension of the pi calculus with function symbols, for instance for tupling and for encryption, that can be assumed to satisfy particular equations. So we first select function symbols and an equational theory for modelling the messages of

JFKr. We also review the syntax and semantics of processes in the applied pi calculus. (We refer to prior work [Abadi and Fournet 2001] for additional definitions, motivations, and examples.) Finally, we introduce a few technical notations and concepts based on the operational semantics of the applied pi calculus.

3.1 An Equational Theory

In general, a *signature* Σ consists of a finite set of function symbols, such as g and H, each with an integer arity. Given a signature Σ, an infinite set of names, and an infinite set of variables, the set of *terms* is defined by the grammar:

$U, V ::=$	terms
$\quad c, d, n, s, \ldots$	name
$\quad x, y, \ldots$	variable
$\quad f(U_1, \ldots, U_l)$	function application

where f ranges over the function symbols of Σ and l matches the arity of f. We use metavariables u and v to range over both names and variables. Furthermore, given a signature Σ, we equip it with an equational theory (that is, with an equivalence relation on terms with certain closure properties). We write $\Sigma \vdash U = V$ when the equation $U = V$ is in the theory associated with Σ. We usually keep the theory implicit, and abbreviate $\Sigma \vdash U = V$ to $U = V$ when Σ is clear from context or unimportant.

For the study of JFK, we pick Σ in such a way that we have the following grammar for terms:

$M, T, U, V ::=$	Terms
$\quad c, d, n, s$	name
$\quad N, K, k, x, y, z$	variable
$\quad \mathsf{E}\{U\}(T)$	shared-key encryption
$\quad \mathsf{D}\{U\}(T)$	shared-key decryption
$\quad \mathsf{H}\{U\}(T)$	keyed cryptographic hash function
$\quad \mathsf{g}$	Diffie-Hellman group
$\quad U\,\hat{}\,V$	Diffie-Hellman exponentiation
$\quad \mathsf{Pk}(U)$	public key (and identity) from private key
$\quad \mathsf{S}\{U\}(T)$	public-key signature
$\quad \mathsf{V}\{U\}(V, T)$	public-key signature verification
$\quad \mathsf{RecoverKey}(V)$	public-key recovery (for the attacker)
$\quad \mathsf{RecoverText}(V)$	text recovery (for the attacker)
$\quad \mathsf{true}$	true
$\quad \mathrm{e}, \mathrm{a}, \mathrm{v}, \mathrm{i}, \mathrm{r}$	constant tags for keyed-hash specialization
$\quad \mathsf{cons}(V_1, V_2)$	pairing
$\quad \mathsf{F}_1^{\mathsf{cons}}(T), \mathsf{F}_2^{\mathsf{cons}}(T)$	projections for pairs
$\quad 1(V_1, V_2), \ldots, 4(V_1, V_2)$	constructors for formatted messages
$\quad \mathsf{F}_1^1(T), \ldots, \mathsf{F}_2^4(T)$	projections for formatted messages
$\quad \emptyset$	empty set
$\quad U.V$	set extension

where we put some arguments in braces only as a syntactic convenience (so, for example, $\mathsf{H}\{U\}(T)$ stands for $\mathsf{H}(U, T)$), and use $\hat{}$ and . as infix function symbols. These terms include names and variables, cryptographic constructs, and auxiliary constructs for tags, pairs, formatted messages, and sets, as follows:

—JFKr uses formatted IP messages, each with a fixed number of fields. Accordingly, we introduce function symbols $1(\_,\_)$, $2(\_,\_,\_,\_,\_)$, $3(\_,\_,\_,\_,\_,\_,\_)$, $4(\_,\_)$ in the signature Σ; these symbols represent the message constructors. In addition, we introduce inverse, unary function symbols $\mathsf{F}_1^1(\_)$ and $\mathsf{F}_2^1(\_)$ to select the fields in Message 1, and similarly for the other messages. Finally, we describe the intended behavior of formatted messages with the evident equations:

$$\mathsf{F}_i^{\mathsf{n}}(\mathsf{n}(x_1, \ldots, x_i, \ldots)) = x_i \quad \text{projections for formatted messages } (\mathsf{n} = 1, 2, 3, 4)$$

Similarly, for pairing, we have the equations:

$$\mathsf{F}_i^{\mathsf{cons}}(\mathsf{cons}(x_1, x_2)) = x_i \quad \text{projections for pairs}$$

When we use formatted messages (rather than pairing and tupling), it is only for clarity: 1, 2, 3, and 4 are convenient tags. It might appear that the use of tags could have security implications, since for instance $1(U, V)$ and $4(U, V)$ cannot be confused, while confusions between messages can sometimes facilitate attacks. However, because terms of the forms $1(\_,\_)$, $2(\_,\_,\_,\_,\_)$, $3(\_,\_,\_,\_,\_,\_,\_)$, and $4(\_,\_)$ are never cryptographically protected, a would-be active attacker can modify the tags 1, 2, 3, 4 at will, so the tags do not provide any protection.

—In order to model symmetric cryptography (that is, shared-key cryptography), we introduce binary function symbols $\mathsf{E}\{\_\}(\_)$ and $\mathsf{D}\{\_\}(\_)$ for encryption and decryption, respectively, with the equation:

$$\mathsf{D}\{y\}(\mathsf{E}\{y\}(x)) = x \quad \text{shared-key decryption}$$

Here x represents the plaintext and y the key. This and other equations embody our (fairly standard) hypotheses on the cryptographic primitives introduced in Section 2.

—It is only slightly harder to model public-key signatures, where the keys for signing and verification are different. In addition to symbols for signing $\mathsf{S}\{\_\}(\_)$ and signature verification $\mathsf{V}\{\_\}(\_,\_)$, we introduce the unary function symbol $\mathsf{Pk}(\_)$ for deriving a public verification key from a private signing key, and the equation:

$$\mathsf{V}\{\mathsf{Pk}(y)\}(\mathsf{S}\{y\}(x), x) = \mathtt{true} \quad \text{public-key signature verification}$$

—In order to model the keyed hash function used in JFK, we introduce a binary function symbol $\mathsf{H}\{\_\}(\_)$ with no equations. The fact that $\mathsf{H}\{K\}(V) = \mathsf{H}\{K'\}(V')$ only when $K = K'$ and $V = V'$ models that H is collision-free. The absence of an inverse for H models the one-wayness of H. In our protocol, these properties are important for guaranteeing, for instance, that keyed hashes h_I, h_R, and K_R cannot be forged.

—More interestingly, exponentiation $\_\,\hat{}\,\_$ has no inverse, but an equation accounts for the commutativity property used for establishing a shared secret.

$$(\mathsf{g}\,\hat{}\,x)\,\hat{}\,y = (\mathsf{g}\,\hat{}\,y)\,\hat{}\,x \quad \text{Diffie-Hellman computation}$$

—Some of the functions and equations are not needed in the protocol itself, but may (in principle) weaken the protocol for the benefit of an attacker; the functions $\mathsf{RecoverKey}(\_)$

and $\mathsf{RecoverText}(\_)$ can be used to extract information from signatures, as specified in the following two equations:

$$\begin{array}{rcll} \mathsf{RecoverKey}(\mathsf{S}\{y\}(x)) & = & \mathsf{Pk}(y) & \text{public-key recovery from a signature} \\ \mathsf{RecoverText}(\mathsf{S}\{y\}(x)) & = & x & \text{signed text recovery from a signature} \end{array}$$

We could further refine our theory by reflecting known weaknesses of the underlying cryptographic algorithms or their interactions, by considering additional equations (for instance refining $\mathsf{H}\{\_\}(\_)$, in the spirit of Abadi and Fournet [2001, Section 6]).

—The equations for the remaining constructs are fairly mundane:

$$\begin{array}{rcll} (\emptyset.x).x & = & \emptyset.x & \text{idempotence of set extension} \\ (x.y).z & = & (x.z).y & \text{commutativity of set extension} \end{array}$$

We have functions for constructing sets, but not a set membership relation; instead, we let $U \in V$ abbreviate $V.U = V$.

3.2 Syntax and Informal Semantics for Processes

The grammar for *processes* in the applied pi calculus is similar to the one in the pi calculus, except that messages can contain terms (rather than only names) and that names need not be just channel names:

$P, Q, R ::=$	processes (or plain processes)
$\mathbf{0}$	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction (“new”)
$\mathit{if}\ U = V\ \mathit{then}\ P\ \mathit{else}\ Q$	conditional
$u(x).P$	message input
$\overline{u}\langle V\rangle.P$	message output

The null process $\mathbf{0}$ does nothing; $P \mid Q$ is the parallel composition of P and Q; the replication $!P$ behaves as an infinite number of copies of P running in parallel. The process $\nu n.P$ makes a new name n then behaves as P. We often use ν as a generator of unguessable seeds. In some cases, those seeds may directly serve as cryptographic keys; in others, some transformations are needed for deriving keys from seeds. The conditional construct $\mathit{if}\ U = V\ \mathit{then}\ P\ \mathit{else}\ Q$ is standard, but we should stress that $U = V$ represents equality, rather than strict syntactic identity. We abbreviate it $\mathit{if}\ U = V\ \mathit{then}\ P$ when Q is $\mathbf{0}$. Finally, the input process $u(x).P$ is ready to input from channel u, then to run P with the actual message replaced for the formal parameter x, while the output process $\overline{u}\langle V\rangle.P$ is ready to output message V on channel u, then to run P. In both of these, we may omit P when it is $\mathbf{0}$. For instance, the (useless) process $\nu K.\overline{c}\langle \mathsf{E}\{K\}(M)\rangle$ sends the term M encrypted under a fresh key K on channel c.

Further, we extend processes with *active substitutions*:

$A, B, C ::=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{x = M\}$	active substitution

We write $\{x = M\}$ for the substitution that replaces the variable x with the term M. The substitution $\{x = M\}$ typically appears when the term M has been sent to the environment, but the environment may not have the atomic names that appear in M; the variable x is just a way to refer to M in this situation. The substitution $\{x = M\}$ is active in the sense that it "floats" and applies to any process that comes into contact with it. In order to control this contact, we may add a variable restriction: we define $let\ \{x = M\}\ in\ P$ as syntactic sugar for $P\{x = M\}$. Although the substitution $\{x = M\}$ concerns only one variable, we can build bigger substitutions by parallel composition. In particular, we can write $let\ \{x_1 = V_1\} \mid \dots \mid \{x_n = V_n\}\ in\ P$, which is P with local variables $x_1, \dots, x_n$ bound to $V_1, \dots, V_n$, respectively. We always assume that our substitutions are cycle-free. We also assume that, in an extended process, there is at most one substitution for each variable, and there is exactly one when the variable is restricted.

Although the syntax draws a formal distinction between ordinary processes and extended processes, we typically ignore this distinction, for simplicity. In particular we often use the symbols P, Q, and R for extended processes, thus avoiding possible confusions between extended processes and principals.

A *frame* is an extended process built up from active substitutions by parallel composition and restriction. Informally, frames represent the static knowledge gathered by the environment after communications with an extended process. We let φ range over frames. The frame associated with an extended process is obtained by erasing its plain process components. A *context* $C[\_]$ is a process with a hole, and $C[P]$ is the result of filling $C[\_]$'s hole with P. An *evaluation context* $C[\_]$ is an extended process with a hole in the place of an extended process. As usual, names and variables have scopes, which are delimited by restrictions and by inputs. When E is any expression, $fv(E)$, $bv(E)$, $fn(E)$, and $bn(E)$ are the sets of free and bound variables and free and bound names of E, respectively.

We rely on a sort system for terms and extended processes [Abadi and Fournet 2001, section 2]. We always assume that terms and extended processes are well-sorted and that substitutions and context applications preserve sorts.

3.3 Syntactic Sugar

In our formalization of JFK, we rely on various abbreviations for processes and data structures. We write $if\ M\ then\ P$ instead of $if\ M = \mathtt{true}\ then\ P$. Given a finite set I and a family of extended processes P_i, one for each $i \in I$, we let $\prod_{i \in I} P_i$ be the parallel composition of the extended processes P_i. We omit pair constructors and parentheses for nested pairs, writing for instance $\mathsf{H}\{K\}(x_R, N_R, N_I)$ for $\mathsf{H}\{K\}(\mathsf{cons}(x_R, \mathsf{cons}(N_R, N_I)))$. We use pattern matching on tuples as syntactic sugar for the corresponding selectors, writing for instance

$$c(1(=N_I, x_I)).P$$

instead of

$$c(z).let\ \{x_I = \mathsf{F}^1_2(z)\}\ in\ if\ z = 1(N_I, x_I)\ then\ P$$

for some fresh variable z; this process receives a message on channel c, matches it with $1(N_I, T)$ for some subterm T, then runs P with T substituted for x_I; otherwise, the received message is silently discarded. We also define syntax for filtering duplicate messages:

$$!c(X)\backslash V.C[if\ T\ fresh\ then\ P]$$

stands for

$$\nu s.(\overline{s}\langle V\rangle \mid !c(X).C[s(z).(\overline{s}\langle z.T\rangle \mid \mathit{if}\ T \notin z\ \mathit{then}\ P)])$$

where $C[\_]$ is a context, X is a pattern, s is a fresh channel name, and z is a fresh variable. We use the local channel s for maintaining a set V of previous values for the term T. The arrival of a message may cause the addition of a particular T (which may depend on variables bound in X) to this set, and the execution of P. For instance, taking $X = T = x$, the process $!c(x)\backslash\emptyset.\mathit{if}\ x\ \mathit{fresh}\ \mathit{then}\ P$ runs P once for every distinct message M received on c, with M substituted for x, and silently drops any duplicate message received on c.

3.4 Operational Semantics

Structural equivalences, written $P \equiv Q$, relate extended processes that are equal by any capture-avoiding rearrangements of parallel compositions, restrictions, and active substitutions, and by equational rewriting of any terms in processes. For example, we have that

$$\nu s.(\overline{c}\langle s\rangle \mid c(x).\overline{d}\langle x\rangle) \equiv c(x).\overline{d}\langle x\rangle \mid \nu s.\overline{c}\langle s\rangle$$

and, if $s = V$ in the underlying equational theory, that

$$\nu s.(\overline{c}\langle V\rangle \mid c(x).\overline{d}\langle x\rangle) \equiv c(x).\overline{d}\langle x\rangle \mid \nu s.\overline{c}\langle s\rangle$$

We also have that

$$\nu s.(\overline{c}\langle \mathsf{E}\{s\}(\mathsf{g})\rangle \mid c(x).\overline{d}\langle x\rangle) \equiv \nu y.(\{y = \mathsf{E}\{s\}(\mathsf{g})\} \mid \nu s.(\overline{c}\langle y\rangle \mid c(x).\overline{d}\langle x\rangle))$$

Reductions and labelled transitions, which we explain next, are closed by structural equivalence, hence by equational rewriting on terms.

Reductions, written $P \rightarrow Q$, represent silent steps of computation (that is, internal message transmissions and branching on conditionals). For example, we have that

$$\nu s.(\overline{c}\langle s\rangle \mid c(x).\overline{d}\langle x\rangle) \rightarrow \nu s.\overline{d}\langle s\rangle$$

Labelled transitions, written $P \xrightarrow{\alpha} Q$, represent interactions between the extended process P and its environment. Specifically, $P \xrightarrow{c(V)} Q$ and $P \xrightarrow{\nu\widetilde{u}.\overline{c}\langle V\rangle} Q$ represent inputs and outputs, respectively, from P's viewpoint. In both, P and Q are extended processes, c is a communication channel, and V is a message. The labels $c(V)$ and $\nu\widetilde{u}.\overline{c}\langle V\rangle$ are the actions of these labelled transitions; an action indicates the nature and contents of an interaction with the environment: whether a communication step is an input or an output, on what channel it takes place, and the corresponding message. In general, output actions $\nu\widetilde{u}.\overline{c}\langle V\rangle$ include restrictions on the fresh names and variables $\widetilde{u}$ that occur in the message in question; after the transition, the environment gains access to $\widetilde{u}$ and may use them to perform further actions. In contrast with other process calculi, an output transition $P \xrightarrow{\nu\widetilde{u}.\overline{c}\langle V\rangle} Q$ is defined only for terms V that do not export restricted names (unless V is a name). Nonetheless, Q may contain an active substitution that associates variables in $\widetilde{u}$ with any terms. For example, we have that

$$\nu s.(\overline{c}\langle s\rangle \mid c(x).\overline{d}\langle x\rangle) \xrightarrow{\nu s.\overline{c}\langle s\rangle} c(x).\overline{d}\langle x\rangle$$

and that

$$\nu s.(\overline{c}\langle \mathsf{E}\{s\}(\mathsf{g})\rangle \mid c(x).\overline{d}\langle x\rangle) \xrightarrow{\nu y.\overline{c}\langle y\rangle} \nu s.(\{y = \mathsf{E}\{s\}(\mathsf{g})\} \mid c(x).\overline{d}\langle x\rangle)$$

An input transition $P \xrightarrow{c(V)} Q$ may use variables defined in P (typically from previous message outputs) to form the message V. For example, we have that

$$\nu s.(\{y = \mathsf{E}\{s\}(\mathsf{g})\} \mid c(x).\overline{d}\langle x\rangle) \xrightarrow{c(y)} \nu s.(\{y = \mathsf{E}\{s\}(\mathsf{g})\} \mid \overline{d}\langle y\rangle)$$

3.5 Observational Equivalences

In the analysis of protocols, we frequently argue that two given processes cannot be distinguished by any context, that is, that the processes are observationally equivalent. As in the spi calculus [Abadi and Gordon 1999], the context represents an active attacker, and equivalences capture security properties in the presence of the attacker. The applied pi calculus has a useful, general theory of observational equivalence parameterized by Σ and its equational theory [Abadi and Fournet 2001]. Specifically, the following three relations are defined for any Σ and equational theory:

—*Static equivalence*, written $\approx_s$, relates frames that cannot be distinguished by any term comparison. In the presence of the ν construct, the relation $\approx_s$ is somewhat delicate and interesting. For instance, we have $\nu N.\{x = \mathsf{H}\{N\}(V)\} \approx_s \nu N.\{x = \mathsf{H}\{N\}(V')\}$ for any terms V and V', since the nonce N guarantees that both terms substituted for x have the same (null) equational properties, but $\nu N.\{x = 1(N, V)\} \not\approx_s \nu N.\{x = 1(N, V')\}$, as soon as V and V' differ, since the comparison $\mathsf{F}^1_2(x) = V$ succeeds only with the first frame. We say that two extended processes are statically equivalent when their associated frames are.

—More generally, *observational equivalence*, written $\approx$, relates extended processes that cannot be distinguished by any evaluation context in the applied pi calculus, with any combination of messaging and term comparisons; this relation is used to state some of our main results on JFKr.

—*Labelled bisimilarity*, written $\approx_l$, coincides with observational equivalence, but it is defined in terms of labelled transitions instead of arbitrary evaluation contexts, and it is the basis for standard, powerful proof techniques.

3.6 Traces and Related Notions

We close this section with a few technical notations and concepts based on the operational semantics of the applied pi calculus. These are not specific to JFK, but we use them in its study.

As discussed in Section 4.1 below, eavesdropping amounts to a message interception followed by a re-emission of the same message. Formally, the interception corresponds to an output label $\nu\widetilde{u}.\overline{c}\langle V\rangle$, and the re-emission corresponds to an input label $c(V)$. We write $P \xrightarrow{\nu\widetilde{u}.c[V]} P'$ as a shorthand for the two transitions $P \xrightarrow{\nu\widetilde{u}.\overline{c}\langle V\rangle}\xrightarrow{c(V)} P'$.

A trace $P \xrightarrow{\eta} Q$ represents a sequence of computation steps, from process P to process Q with the sequence of labels η, possibly interleaved with internal computation steps, which are kept implicit. In what follows, and especially when relying on automated proofs, we often consider a particular class of traces. For a given sequence of labels η, we say that the trace $P \xrightarrow{\eta} Q$ is *normal* when:

(1) The free names and variables of P and the extruded names and variables of η do not clash. (Such clashes can be prevented by renaming names and variables.) Hence, there

can be a binding "ν" in front of an x or n in an output of η only when x or n does not appear free in any process or transition preceding that output.

(2) There is no internal communication step on any free channel. (This condition does not entail any loss of generality, since every communication step on a free channel can be represented as a series of two transitions, an output immediately followed by an identical input, as in our model of eavesdropping.)

Those conditions are technically convenient, but not essential.

We also often compare actions in traces up to equational rewriting. For any trace $P \xrightarrow{\eta} Q$, we have $Q \equiv \nu\widetilde{n}.(\varphi \mid R)$ where $\equiv$ is the structural-equivalence relation, R is a process with no active substitutions, and $\nu\widetilde{n}.\varphi$ is a frame associated with Q that defines the variables exported in η. Assuming that $P \xrightarrow{\eta} Q$ is normal and that the names $\widetilde{n}$ do not occur free in P or in η, we can disregard the location of restrictions in labels, and consider labels up to equality under the substitution φ. This comparison "after η" depends on Q, but not on a particular choice for φ. Informally, it represents term comparison as observed from R.

4. A MODEL OF JFK IN THE APPLIED PI CALCULUS

Next we discuss our representations for the IP network, attackers, and principals, and we assemble processes that represent configurations of principals.

4.1 The Network and the Attacker

In our model, all IP messages are transmitted on a free pi calculus communication channel, c, which represents a public IP network in which message contents serve for differentiating traffic flows. An arbitrary environment (an arbitrary evaluation context) represents the attacker. This environment can interact with other principals by inputs and outputs on any free channel, including c.

As a special case, we sometimes consider a weaker, passive attacker that only eavesdrops on messages but does not modify them. An attack step against a process P consists in eavesdropping on a message sent by P, and amounts to a message interception followed by a re-emission of the same message.

4.2 Configurations of Principals

Our model allows an arbitrary number of principals. Each principal may run any number of sessions, as initiator and as responder, and may perform other operations after session establishment or even independently of the protocol. Only some of these principals follow the protocol. We are interested in the security properties that hold for them.

For the present purposes, the essence of a principal lies in its ability to produce signatures verifiable with its public key. Accordingly, we refer to each principal by its public key, using variables ID_A, ID_B,... for both identities and public keys. We also associate the context

$$PK^A[\_] = \nu K_-^A.(\{\mathsf{ID}_A = \mathsf{Pk}(K_-^A)\} \mid [\_])$$

with every principal A. This context restricts the use of the signing key K_-^A to the process in the context and it exports the corresponding verification key ID_A. Since there is no inverse for $\mathsf{Pk}(\_)$, the verification key can be passed to the environment without giving away the capability to sign messages with K_-^A to the environment. Whenever we put a process R in this context, our intent is that R never communicates K_-^A to the environment. By definition of well-formed configurations in the applied pi calculus, a process of the form $Q \mid PK^A[R]$ exports ID_A; both Q and R can access ID_A; only R can access K_-^A; and no context may change the binding of ID_A to $\mathsf{Pk}(K_-^A)$. On the other hand, the context may define any number of other principals. Thus, we obtain a fairly generous and convenient model when we represent an attacker by an arbitrary context.

We let $\mathcal{C}$ range over sets of compliant principals—that is, principals that entirely delegate the use of their signing keys to JFKr. While some properties will obviously hold only for compliant principals, the initiator and responder code do not assume knowledge of $\mathcal{C}$: indeed, compliant and non-compliant principals can attempt to establish sessions.

Compliant principals rely on an implementation of JFKr, written as a process $\mathcal{S}$. (The notation $\mathcal{S}$ stands for "system".) In addition, each compliant principal A has a "user process", written $\mathcal{P}^A$. The user process defines any additional behavior, such as when protocol runs are initiated and what happens to the shared secret K_v after each session establishment. While we define $\mathcal{S}$ below, we treat $\mathcal{P}^A$ as an abstract parameter, in the context that encloses $\mathcal{S}$, possibly under the control of the attacker. Each user process interacts with $\mathcal{S}$ through the following control interface:

—As initiator, $\mathcal{P}^A$ sends a message $\overline{\mathit{init}}^A\langle \mathsf{ID}'_R, \mathsf{sa}_I\rangle$ to initiate a new session, with responder hint ID'_R and security association sa_I. When the protocol completes successfully, $\mathcal{S}$ sends $\overline{\mathit{connect}}^A\langle \mathsf{ID}_B, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ to notify $\mathcal{P}^A$ that the session has been accepted, and that A now shares K_v with a principal with identifier ID_B.

—As responder, $\mathcal{S}$ sends $\overline{\mathit{accept}}^A\langle \mathsf{ID}_B, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ to notify $\mathcal{P}^A$ that it has accepted a session initiated by a principal with identifier ID_B, parameters ID'_R, sa_I, sa_R and shared secret K_v. To control who can initiate a session with A, $\mathcal{S}$ is parameterized by a set S_I^A of acceptable initiator identities. (We do not need a set such as S_I^A at the initiator: after completion of the protocol, the initiator's user process can decide what to do with the new session depending on the responder identity in the *connect* message.) For simplicity, S_I^A and sa_R are fixed; we also assume that these terms do not contain any name or variable restricted in $\mathcal{S}$.

Thus, the interface between each principal A and JFKr consists of three communication channels init^A, accept^A, and $\mathit{connect}^A$ plus a set of identities S_I^A. The channels init^A, accept^A, and $\mathit{connect}^A$ can be restricted (with ν) in order to hide the interface from the environment.

For example, the user process $\mathcal{P}^A$ of a principal A may be:

$$\overline{\mathit{init}}^A\langle \mathsf{ID}_B, \mathsf{sa}_I\rangle.\mathit{connect}^A(=\mathsf{ID}_B, =\mathsf{ID}_B, =\mathsf{sa}_I, \mathsf{sa}_R, K_v).\overline{c}\langle \mathsf{E}\{K_v\}(\mathsf{true})\rangle$$

This code initiates a single session, specifically with B; after connecting, it sends the term true encrypted under the resulting session key K_v on the channel c that represents the public IP network. A variant of this code can also accept a session and then send the term true encrypted under the resulting session key K_v on c:

$$\begin{array}{l}\overline{\mathit{init}}^A\langle \mathsf{ID}_B, \mathsf{sa}_I\rangle.\mathit{connect}^A(=\mathsf{ID}_B, =\mathsf{ID}_B, =\mathsf{sa}_I, \mathsf{sa}_R, K_v).\overline{c}\langle \mathsf{E}\{K_v\}(\mathsf{true})\rangle \\ \mid \mathit{accept}^A(=\mathsf{ID}_B, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v).\overline{c}\langle \mathsf{E}\{K_v\}(\mathsf{true})\rangle\end{array}$$

Another principal B may have a similar user process $\mathcal{P}^B$, for example:

$$\mathit{accept}^B(=\mathsf{ID}_A, =\mathsf{ID}_B, \mathsf{sa}_I, \mathsf{sa}_R, K_v).c(x).\overline{c}\langle \mathsf{D}\{K_v\}(x)\rangle$$

Putting $\mathcal{P}^A$ and $\mathcal{P}^B$ together with $\mathcal{S}$, and letting the set $\mathcal{C}$ of compliant principals be $\{A, B\}$, we obtain a configuration $\mathcal{P}^A \mid \mathcal{P}^B \mid \mathcal{S}$. Provided that $A \in S_I^B$, this sample configuration can execute one run of JFKr. We may also add restrictions on the channels $init^A$, $accept^A$ $connect^A$, $init^B$, $accept^B$, and $connect^B$, thus ensuring that only $\mathcal{P}^A$ and $\mathcal{P}^B$ control $\mathcal{S}$.

4.3 The Protocol

Figure 2 shows our implementation of JFKr in the applied pi calculus. It includes definitions of processes for each role: a single process (I_0^A) for the initiator, and two processes (R_1^A, R_3^A) that do not share session state for the responder. For each principal A, these replicated processes perform tests on incoming messages and compute outgoing messages. (Here, we give processes for A in both roles; elsewhere, when describing an exchange between two principals, we often use A as initiator and B as responder.) The code of I_0^A, R_1^A, and R_3^A represents the steps in the informal protocol narration of Section 2.1. Comments in the figure explain the code. As further explanation, we paraphrase the code of R_1^A, as an example. This code starts with the reception of a message $1(N_I, x_I)$ on the public IP network. The reception is replicated, because this code is expected to run whenever it is triggered by an input; several instances of the code may execute in parallel. Then the code generates a fresh nonce N_R and computes an anti-DOS cookie t_R. Finally, it sends $2(N_I, N_R, x_R, \mathsf{g}_R, t_R)$ on the public IP network.

The figure also includes the definition of a configuration $\mathcal{S}$: an assembly of an arbitrary but fixed set of compliant principals $\mathcal{C}$ that potentially share an arbitrary but fixed pool of exponentials X. We always assume that $\mathcal{C}$ and X are not empty.

The design of JFK allows reusing Diffie-Hellman exponents for several sessions, principals, and roles, and does not impose a particular policy for changing them. For each exponent, one can decide when to stop using that exponent in new sessions. For instance, an exponent may expire once the first session established using that exponent terminates, so that discarding session keys prevents their later compromise. In our model, all compliant principals may use any number of shared exponentials, in both roles, for any number of parallel sessions. Results for configurations with less sharing are immediate corollaries of ours.

The context $D_x[\_]$ represents a Diffie-Hellman party, d_x the corresponding secret exponent, x the derived exchange value (the exponential), and g the group (the same one for all compliant principals). The set X contains the exponentials shared by the compliant principals. The context $D_X[\_]$ consists of contexts $D_x[\_]$ for each $x \in X$. For simplicity, according to the code, compliant principals never disclose exponents.

In contrast with actual implementations of JFK, our model treats abstractly several aspects of the protocol. In particular, it uses an unambiguous format for all messages, thereby assuming, for instance, that the wire format for messages does not leak additional information, and that ill-formed messages are safely ignored. (In our model, ill-formed messages cause pattern matching or other tests to fail, thereby discarding the message and, except for Message 3, aborting the session.) Furthermore, it does not cover IP addressing, routing, and fragmentation concerns, the contents of the security-association parameters sa_z, the handling of ID'_R, the potential usage of several groups g, aspects of caching, and error messages. We made such simplifications partly by choice, partly by necessity; the resulting model remains quite informative and rich.

$$
\begin{array}{rcll}
I_0^A & = & !\mathit{init}^A(\mathsf{ID}'_R, \mathsf{sa}_I). & \textbf{Initiator for each message } \textbf{\textit{init}} \\
 & & \nu N_I. & \quad \textit{create a fresh nonce} \\
 & & \overline{c}\langle \mathsf{1}(N_I, x_I)\rangle. & \quad \textit{send Message 1} \\
 & & c(\mathsf{2}(=N_I, N_R, x_R, \mathsf{g}_R, t_R)). & \quad \textit{wait for Message 2} \\
 & & \mathit{let}\ \kappa_I\ \mathit{in} & \quad \textit{compute shared keys (see below)} \\
 & & \mathit{let}\ \{s_I = \mathsf{S}\{K_-^A\}(N_I, N_R, x_I, x_R, \mathsf{g}_R)\}\ \mathit{in} & \quad \textit{sign} \\
 & & \mathit{let}\ \{e_I = \mathsf{E}\{K_e\}(\mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, s_I)\}\ \mathit{in} & \quad \textit{encrypt} \\
 & & \mathit{let}\ \{h_I = \mathsf{H}\{K_a\}(\mathsf{i}, e_I)\}\ \mathit{in} & \quad \textit{compute MAC} \\
 & & \overline{c}\langle \mathsf{3}(N_I, N_R, x_I, x_R, t_R, e_I, h_I)\rangle. & \quad \textit{send Message 3} \\
 & & c(\mathsf{4}(e_R, h_R)). & \quad \textit{wait for Message 4} \\
 & & \mathit{if}\ \mathsf{H}\{K_a\}(\mathsf{r}, e_R) = h_R\ \mathit{then} & \quad \textit{check MAC} \\
 & & \mathit{let}\ \{\mathsf{ID}_R, \mathsf{sa}_R, s_R = \mathsf{D}\{K_e\}(e_R)\}\ \mathit{in} & \quad \textit{decrypt} \\
 & & \mathit{if}\ \mathsf{V}\{\mathsf{ID}_R\}(s_R, (N_I, N_R, x_I, x_R))\ \mathit{then} & \quad \textit{check signature} \\
 & & \overline{\mathit{connect}}^A\langle \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle & \quad \textit{complete keying}
\end{array}
$$

$$
\begin{array}{rcll}
R_1^A & = & !c(\mathsf{1}(N_I, x_I)). & \textbf{Responder for each Message 1} \\
 & & \nu N_R. & \quad \textit{create a fresh nonce} \\
 & & \mathit{let}\ \{t_R = \mathsf{H}\{K_R\}(x_R, N_R, N_I)\}\ \mathit{in} & \quad \textit{compute anti-DOS token} \\
 & & \overline{c}\langle \mathsf{2}(N_I, N_R, x_R, \mathsf{g}_R, t_R)\rangle & \quad \textit{send Message 2} \\
 & & & \\
R_3^A & = & !c(\mathsf{3}(N_I, N_R, x_I, x, t_R, e_I, h_I))\backslash\emptyset. & \textbf{Responder for each Message 3} \\
 & & \mathit{if}\ t_R = \mathsf{H}\{K_R\}(x, N_R, N_I)\ \mathit{then} & \quad \textit{check anti-DOS token} \\
 & & \mathit{if}\ t_R\ \mathit{fresh\ then} & \quad \textit{accept token only once} \\
 & & \prod_{x_R \in X} \mathit{if}\ x = x_R\ \mathit{then} & \quad \textit{branch on DH exponential} \\
 & & \mathit{let}\ \kappa_R\ \mathit{in} & \quad \textit{compute shared keys (see below)} \\
 & & \mathit{if}\ \mathsf{H}\{K_a\}(\mathsf{i}, e_I) = h_I\ \mathit{then} & \quad \textit{check MAC} \\
 & & \mathit{let}\ \{\mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, s_I = \mathsf{D}\{K_e\}(e_I)\}\ \mathit{in} & \quad \textit{decrypt} \\
 & & \mathit{if}\ \mathsf{ID}_I \in S_I^A\ \mathit{then} & \quad \textit{authorize} \\
 & & \mathit{if}\ \mathsf{V}\{\mathsf{ID}_I\}(s_I, (N_I, N_R, x_I, x_R, \mathsf{g}_R))\ \mathit{then} & \quad \textit{check signature} \\
 & & \overline{\mathit{accept}}^A\langle \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle. & \quad \textit{accept the session} \\
 & & \mathit{let}\ \{s_R = \mathsf{S}\{K_-^A\}(N_I, N_R, x_I, x_R)\}\ \mathit{in} & \quad \textit{sign} \\
 & & \mathit{let}\ \{e_R = \mathsf{E}\{K_e\}(\mathsf{ID}_A, \mathsf{sa}_R, s_R)\}\ \mathit{in} & \quad \textit{encrypt} \\
 & & \mathit{let}\ \{h_R = \mathsf{H}\{K_a\}(\mathsf{r}, e_R)\}\ \mathit{in} & \quad \textit{compute MAC} \\
 & & \overline{c}\langle \mathsf{4}(e_R, h_R)\rangle & \quad \textit{send Message 4}
\end{array}
$$

$$
\begin{array}{rcll}
\mathcal{S} & = & D_X\left[\prod_{A \in \mathcal{C}} PK^A\left[I^A | R^A\right]\right] & \textbf{Compliant principal configuration} \\
I^A & = & \prod_{x_I \in X} I_0^A & \quad \textit{A as initiator} \\
R^A & = & \nu K_R.(\prod_{x_R \in X} R_1^A \mid R_3^A) & \quad \textit{A as responder} \\
 & & & \\
PK^A[\_] & = & \nu K_-^A.(\{\mathsf{ID}_A = \mathsf{Pk}(K_-^A)\} \mid [\_]) & \quad \textit{A's signing and verification keys} \\
 & & & \\
D_x[\_] & = & \nu d_x.(\{x = \mathsf{g}\,\hat{}\,d_x\} \mid [\_]) & \quad \textit{DH secret } d \textit{ and exchange value } x \\
D_X[\_] & = & D_{x_1}[\ldots D_{x_n}[\_]] \text{ where } X = \{x_1, \ldots, x_n\} & \quad \textit{shared exponentials} \\
\kappa_I & = & \prod_{u=a,e,v}\{K_u = \mathsf{H}\{x_R\,\hat{}\,d_{x_I}\}(N_I, N_R, \mathsf{u})\} & \quad \textit{key computations for I} \\
\kappa_R & = & \prod_{u=a,e,v}\{K_u = \mathsf{H}\{x_I\,\hat{}\,d_{x_R}\}(N_I, N_R, \mathsf{u})\} & \quad \textit{key computations for R}
\end{array}
$$

Fig. 2. JFKr in the applied pi calculus

5. PROVERIF AND ITS USE

In this section, we briefly describe ProVerif and explain how we use it in the analysis of JFK. In particular, we outline our interpretation of the outcome of automated script verifications as proofs of trace properties for JFK configurations expressed in the pi calculus.

5.1 ProVerif and its Extensions for JFK

ProVerif was first designed for proving secrecy properties, which mean that the adversary cannot compute certain values [Blanchet 2001; Abadi and Blanchet 2005a]. ProVerif was then extended for proving correspondence assertions of the form: if some event e is executed, then some events $e_1, \ldots, e_n$ must have been executed before [Blanchet 2002; Abadi and Blanchet 2005b]. Also treated were injective correspondences, which furthermore require that if the event e is executed m times, then the corresponding events $e_1, \ldots, e_n$ must have been executed at least m times. More recently, ProVerif was extended for proving observational equivalences of the form $\nu a_1, \ldots, a_n.P\sigma \approx \nu a_1, \ldots, a_n.P\sigma'$ where σ and σ' are any substitutions that map the free variables of P to terms in a given set [Blanchet 2004], as well as observational equivalences between processes that differ only in the terms they contain [Blanchet et al. 2005].

Further extensions were added for analyzing JFK:

(1) In previous versions, the events added in the protocol description were either "begin" or "end" events, and the correspondence assertions were always of the form "if some end event has been executed, then some begin events must have been executed". As a result, for proving different properties, we often had to modify events in the protocol description. Now, the protocol description contains only one kind of event, and ProVerif determines automatically from the property of interest which events should be "begin", which should be "end", which should be both, and which should simply be ignored.

(2) In previous versions, for a given "end" event, ProVerif returned a set of Horn clauses from which the user could infer which "begin" events must be executed to execute the "end" event. While this mode is still available, we have introduced a rich specification language for correspondence assertions, and ProVerif can tell the user whether a given correspondence property is proved or not, without manual inspection of the clauses.

(3) We have added an optimization that yields dramatic speedups when the property in question contains many events. More specifically, ProVerif now removes redundant hypotheses from Horn clauses when they contain "begin" events: if the clause is of the form $H \wedge H' \rightarrow C$ and there exists σ such that $H\sigma \subseteq H'$ and σ does not operate on variables of H' and C, then ProVerif replaces the clause with the equivalent clause $H' \rightarrow C$. This transformation considerably speeds up the subsumption test for clauses.

(4) We have added support for scenarios with several phases, such as publishing secret keys after the end of the execution of some sessions of the protocol. This extension has been used in proving perfect forward secrecy properties of JFK.

(5) We have extended the treatment of equations proposed by Blanchet [2001] for Diffie-Hellman to more general equations [Blanchet et al. 2005]. This extension allows us to represent encryption and signatures using equations rather than destructors.

5.2 Scripts for Proof Automation

We rely at least partially on ProVerif in most proofs. For that purpose, we code JFK configurations ($\mathcal{S}$ in Figure 2) in the input syntax of ProVerif, which is an ASCII syntax for the applied pi calculus, then we specify the properties to prove and simply run ProVerif. Additional justifications and details on ProVerif appear in Appendix B.

As noted in the introduction, our ProVerif scripts are available at `http://www.di.ens.fr/~blanchet/crypto/jfk.html`. The script for JFKr differs superficially from $\mathcal{S}$: whereas configurations $\mathcal{S}$ are parameterized by fixed sets of compliant principals and shared exponents, the script gives an interface to the adversary that enables the creation of compliant principals (and provides their identities and control interfaces) and of shared exponents (and provides their exponentials). These unfoldings are best omitted in the statements of theorems. For a given configuration $\mathcal{S}$, one can apply an evaluation context to the process defined in the script so that the resulting process becomes observationally equivalent to $\mathcal{S}$ after exporting the exponentials, the principal identities, and the control channels init^A, accept^A, and $\mathit{connect}^A$. A lemma in Appendix B justifies this transformation.

6. RESISTANCE TO DENIAL-OF-SERVICE ATTACKS

In our formal analysis, we first consider the security mechanisms at the early stages of the protocol, before mutual authentication. These mechanisms aim at hardening JFK against certain DOS attacks relevant in IP security (see e.g., Karn and Simpson [1999]). Our formal analysis relies on an understanding of the costs incurred at these stages (much as in Meadows's cost-based framework [2001]). This understanding is based on discussions in the protocol specification. We characterize the occurrences of operations deemed expensive, without a formal measure of their cost.

In JFK, protocol-based DOS is a concern mostly for the responder. By design, until the computation of κ_R, the processing of Messages 1 and 3 is fast and involves almost no state. From this point, the protocol performs CPU-intensive operations (including a Diffie-Hellman exponentiation and two public-key operations), and allocates some session state.

Since in general, in any protocol, the processing of a message may depend on the contents of previously received messages, each principal may maintain some local state for each session of a protocol. This state can be problematic for servers that are willing to start a session whenever they receive a first client message, before adequate authentication. Indeed, an attacker may send (or redirect) first-message traffic to the server, filling its buffers, and eventually causing valid session attempts to be dropped. This concern motivates a common protocol transformation: instead of keeping state for every session in progress, one or both parties MAC (or encrypt) the state, append the result to outgoing messages, and check (or decrypt) the corresponding values in later incoming messages before processing them. Next, we show that this transformation is correct (i.e., preserves equivalence) for a general class of protocols coded as processes.

We relate a sequential implementation of a protocol to a more complex but stateless implementation, using the observational-equivalence relation, $\approx$. This relation is closed by application of evaluation contexts, which can represent active attackers.

LEMMA 1. *Let $C[\_]$ be a context that binds at most the variable x_2, let K_R be a fresh* *name, let* $P = !c(z)$*, and*

$$\begin{aligned} R_2^\circ &= \nu N.\nu t.\overline{c}\langle M_2\rangle.R_3^\circ \\ R_3^\circ &= ?c(3(=t,=N,=x_2,x_3)).R_4 \end{aligned}$$

$$\begin{aligned} R_2 &= \nu N.let\ \{t = \mathsf{H}\{K_R\}(N,x_2)\}\ in\ \overline{c}\langle M_2\rangle \\ R_3 &= !c(3(t,N,x_2,x_3))\backslash\emptyset.if\ t = \mathsf{H}\{K_R\}(N,x_2)\ then\ if\ t\ fresh\ then\ R_4 \end{aligned}$$

We have $C[R_2^\circ] \mid P \approx \nu K_R.(C[R_2] \mid R_3) \mid P$.

In the statement of the lemma, R_2° and R_3° define the sequential implementation of the protocol, whereas R_2 and R_3 define its stateless implementation. In R_2°, we rely on syntactic sugar for pattern-matching with a retry until a message that matches the pattern X is received, writing

$$?c(X).R \qquad \text{for} \qquad \nu l.\big(\overline{l}\langle\rangle \mid !c(X).l().R\big)$$

where l does not occur in X or R. (In the ProVerif scripts, we use instead the more verbose encoding $\nu l.(!c(X).\overline{l}\langle\widetilde{x}\rangle \mid l(\widetilde{x}).R)$, where $\widetilde{x}$ collects the variables bound in X. The two encodings are equivalent, but the latter encoding facilitates automated proofs.) The process R_2 is triggered each time a first message of the protocol is received; the pair N, x_2 represents the state of the protocol at the end of R_2 that is used later in R_4; M_2 represents a second message of the protocol carrying (at least) N and t, and x_3 represents new data received in Message 3. The presence of the same state (N, x_2) in the message received in R_3 is checked using the authenticator t. The inclusion of a fresh nonce N guarantees that all generated authenticators are different. (In R_2°, the generation of a fresh t and the matching $=t$ do not serve any functional purpose; they are performed only so that the two implementations of the protocol behave similarly.) The additional process P is necessary to account for the possibility of receiving any message z and discarding it after a failed test.

The proof of Lemma 1 appears in Appendix A; it relies on standard bisimulation techniques. The lemma is reminiscent of classical replication laws in process calculi, such as $!(Q_2 \mid !Q_3) \approx !Q_2 \mid !Q_3$, since R_2° and R_3 contain replications and $C[\_]$ typically will.

Our next lemma applies this protocol transformation to JFKr. It relates our main model $\mathcal{S}$ (see Figure 2), which features a stateless responder till reception of a Message 3 with a valid token, to a simplified, linear model $\mathcal{S}^\circ$. The lemma enables us to prove properties of JFKr preserved by $\approx$ (such as trace properties) on $\mathcal{S}^\circ$ instead of $\mathcal{S}$.

LEMMA 2. *We have* $\mathcal{S}^\circ \approx \mathcal{S}$*, where* $\mathcal{S}^\circ$ *is* $\mathcal{S}$ *after replacing each* R^A *by*

$$\begin{aligned} R^{\circ A} = {} & \textstyle\prod_{x_R \in X} \\ & !c(1(N_I, x_I)). \\ & \nu N_R, t_R. \\ & \overline{c}\langle 2(N_I, N_R, x_R, \mathsf{g}_R, t_R)\rangle. \\ & ?c(3(=N_I,=N_R,x_I,=x_R,=t_R,e_I,h_I)). \\ & let\ \kappa_R\ in\ \ldots \qquad \textit{(as in } R_3^A\textit{)} \end{aligned}$$

The lemma is essentially a corollary of Lemma 1; its proof appears in Appendix A.

Our next theorem expresses that the responder commits session-specific resources only once an initiator has established round-trip communication, that is, sent a Message 1, received a Message 2, and returned a Message 3 with matching nonces (Property 2). This property helps because the responder controls the emission of tokens and can cheaply invalidate old ones by rekeying K_R, and because a "blind" attacker (weaker than a typical Needham-Schroeder attacker [1978]) may send Messages 1 with fake IP addresses, but then may not be able to eavesdrop on the corresponding Messages 2. The theorem also includes a similar guarantee for the initiator (Property 1).

THEOREM 1 (PROTECTION FROM DOS). *Let $A \in \mathcal{C}$.*

(*1*) *Let $\mathcal{S}_\$$ be $\mathcal{S}$ with an additional output $\overline{\$}\langle N_I \rangle$ before the Diffie-Hellman computation of κ_I in I_0^A.*
For any normal trace $\mathcal{S}_\$ \xrightarrow{\eta} \mathcal{S}'$, each output $\overline{\$}\langle N_I \rangle$ in η is preceded by distinct, successive actions that match init$^A(_,_)$, $\overline{c}\langle 1(N_I,\_)\rangle$, and $c(2(N_I,\_,\_,\_,\_))$.

(*2*) *Let $\mathcal{S}_\$$ be $\mathcal{S}$ with an additional output $\overline{\$}\langle N_I, N_R \rangle$ before the Diffie-Hellman computation of κ_R in R_3^A.*
For any normal trace $\mathcal{S}_\$ \xrightarrow{\eta} \mathcal{S}'$, each output $\overline{\$}\langle N_I, N_R \rangle$ in η is preceded by distinct, successive actions that match $c(1(N_I,\_))$, $\overline{c}\langle 2(N_I, N_R,\_,\_,\_)\rangle$, and $c(3(N_I, N_R,\_,\_,\_,\_,\_))$.

The additional outputs on \$ serve as markers for the start of expensive processing (public-key operations and session-state allocation). The theorem formulates "round-trip authentication" as injective correspondences between actions. Property 1 is almost obvious for JFKr, as the initiator I_0^A sequentially processes all messages and receives a single Message 2 for each session. Property 2 is more interesting and depends on the authenticator, but its proof becomes easy after applying a variant of Lemma 2 to obtain an equivalent, linear protocol. (As discussed in Section 2.3, many Messages 3 may be processed for the same authenticator in incorrect interpretations of JFKr, for which Property 2 is false.) The proof of the theorem appears in Appendix B.

7. CORE SECURITY PROPERTIES: SECRECY AND AUTHENTICITY

Next, we consider session-key secrecy and mutual authentication. We establish fundamental secrecy and authenticity results. Then, more briefly, we discuss perfect forward secrecy and identity-protection properties, which partly follow from those fundamental results.

7.1 Secrecy and Authenticity

Let $\mathcal{S}$ be a JFKr configuration with compliant principals $\mathcal{C}$ sharing exponentials X. We study arbitrary runs of the protocol by examining transitions $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$, where η is an arbitrary sequence of labels. In these labelled transitions, as usual, we omit internal steps $\rightarrow$. Informally, $\mathcal{S}'$ represents any reachable state of the configuration in the presence of an attacker that controls both the low-level IP network (c) and the control interfaces for the principals in $\mathcal{C}$.

The following theorem characterizes runs of the protocol that involve two compliant principals, A and B, in terms of what can be observed by an eavesdropper. We write $\xrightarrow{[1,2,3]}$ for the eavesdropped communications

$$\xrightarrow{\nu N_I.[1(N_I,x_I)]}\xrightarrow{\nu N_R\, t_R.[2(N_I,N_R,x_R,g_R,t_R)]}\xrightarrow{\nu e_I\, h_I.[3(N_I,N_R,x_I,x_R,t_R,e_I,h_I)]}$$

and similarly write $\xrightarrow{[4]}$ for $\xrightarrow{\nu e_R\, h_R.[4(e_R,h_R)]}$. We also write φ_3 and φ_4 for the frames that map the variables N_I, N_R, t_R, e_I, h_I and N_I, N_R, t_R, e_I, h_I, e_R, h_R, K_v, respectively, to distinct restricted names. These frames represent the simplified "net effect" of the runs $\xrightarrow{[1,2,3]}$ and $\xrightarrow{[1,2,3]}\xrightarrow{[4]}$ (including the passing of K_v). Next we examine sessions between compliant principals starting from any reachable state $\mathcal{S}'$ of the protocol.

THEOREM 2 (SECRECY FOR COMPLETE SESSIONS). *For any normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$, principals $A, B \in \mathcal{C}$, exponentials $x_I, x_R \in X$, and terms $\mathsf{ID}'_R, \mathsf{sa}_I$, there exists $\mathcal{S}_3$ such that*

$$\mathcal{S}' \xrightarrow{init^A(\mathsf{ID}'_R,\mathsf{sa}_I)}\xrightarrow{[1,2,3]} \mathcal{S}_3$$

and either (i) $\mathsf{ID}_A \in S_I^B$ and

$$\mathcal{S}_3 \xrightarrow{\nu K_v.\overline{accept}^B\langle \mathsf{ID}_A,\mathsf{ID}'_R,\mathsf{sa}_I,\mathsf{sa}_R,K_v\rangle}\xrightarrow{[4]}\xrightarrow{\overline{connect}^A\langle \mathsf{ID}_B,\mathsf{ID}'_R,\mathsf{sa}_I,\mathsf{sa}_R,K_v\rangle} \approx \mathcal{S}' \mid \varphi_4$$

or (ii) $\mathsf{ID}_A \notin S_I^B$ and $\mathcal{S}_3 \approx \mathcal{S}' \mid \varphi_3$.

The proof of these properties is given in Appendix C; it relies on an analysis of the configurations $\mathcal{S}'$ and on auxiliary equivalences established by ProVerif.

Theorem 2 first expresses the functioning of the protocol, with two normal outcomes depending on $\mathsf{ID}_A \in S_I^B$; the first disjunct is for acceptance, the second for rejection. The two outcomes are not observationally equivalent, informally because an attacker that observes network traffic may be able to tell whether a session succeeds or fails. The theorem also uses observational equivalence to give a simple, abstract characterization of the protocol outcomes: we are (apparently) back to the state of the protocol just before the session began, $\mathcal{S}'$, except for φ_3 and φ_4 which export variables bound to distinct, plain names ($\nu N.\{x = N\}$), our representation of independent, fresh values in the pi calculus. Hence, from the viewpoint of an attacker that can eavesdrop on c and communicate on control interfaces, the intercepted message fields and the session key appear to be fresh, independent names, rather than computed values. In particular, the attacker can learn K_v only through the control interfaces, and e_I and e_R leak nothing about their encrypted contents. Furthermore, the equivalences ensure that the session does not depend on (or affect) any other session in $\mathcal{S}'$. Although the statement of the theorem deals only with a (temporarily) passive attacker, its combination with Theorem 4 (below) does cover all cases of complete sessions.

We also have complementary authentication properties, expressed as correspondence properties on control actions (that is, messages on the control interfaces), now with an active attacker.

THEOREM 3 (AUTHENTICITY FOR CONTROL ACTIONS). *For any normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$, the actions appearing in η have the following properties:*

(1) *For each $\overline{accept}^B\langle \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$, we have $\mathsf{ID}_A \in S_I^B$ and, if $A \in \mathcal{C}$, there is a distinct $init^A(\mathsf{ID}'_R, \mathsf{sa}_I)$.*

(2) *For each $\overline{connect}^A\langle \mathsf{ID}_B, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ there is a distinct $init^A(\mathsf{ID}'_R, \mathsf{sa}_I)$ and, if $B \in \mathcal{C}$, there is a distinct $\overline{accept}^B\langle \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$.*

The proof of these properties partly relies on ProVerif (see Appendix B). For Property 1, we analyze the linear variant of JFKr, then extend the result to JFKr by Lemma 2; in contrast, the direct automated analysis of JFKr yields only a weaker, non-injective correspondence, because ProVerif does not keep track of the linearity enforced by the authenticator cache. ProVerif also yields proofs of these properties for variants of the protocol—with or without sharing of exponentials, for JFKr and for JFKi.

The next theorem also deals with an active attacker. It says that, whenever a normal trace includes a control action $\overline{connect}^A\langle \mathsf{ID}_B, \ldots\rangle$ for some $A, B \in \mathcal{C}$, the trace essentially contains a complete, successful run of the protocol, as described in Theorem 2.

THEOREM 4 (AUTHENTICITY FOR COMPLETE SESSIONS). *Let $A, B \in \mathcal{C}$ and assume that we have a normal trace*

$$\mathcal{S} \xrightarrow{\eta} \xrightarrow{\overline{connect}^A\langle \mathsf{ID}_B, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle} \mathcal{S}'$$

(*1*) $\xrightarrow{\eta}$ *contains a series of transitions that match*

$$\xrightarrow{init^A(\mathsf{ID}'_R, \mathsf{sa}_I)} \xrightarrow{[1,2,3]} \xrightarrow{\nu K_v.\overline{accept}^B\langle \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle} \xrightarrow{[4]}$$

in the same order, except possibly for argument x_I in the first input on c and for argument t_R in the second input and third output on c.

(*2*) *Let η' be η after erasure of these transitions. We have $\mathcal{S} \mid \varphi_4 \xrightarrow{\eta'}\approx \mathcal{S}'$.*

We rely on ProVerif for establishing the first point of this theorem, via Lemma 2 (see Appendix B), and on an analysis of the normal trace under consideration for establishing the second point of the theorem (see Appendix C).

Theorem 3 is simpler and more abstract than Theorem 4, as it deals only with the interface of the protocol, through control actions. Theorem 4 is more complex, as it expresses properties on both control actions and lower-level IP messages exchanged by the protocol. These properties imply that certain protocol inputs match previous protocol outputs, so these inputs are authentic. In general, we would not expect an exact match of all message fields (even if such matches facilitate a formal analysis): some fields are not authenticated. Here, the absence of authentication of x_I in the first message weakens identity protection; see Section 2.3. The absence of authentication of t_R by the initiator seems harmless, inasmuch as t_R is used only by R.

7.2 Perfect Forward Secrecy

As a corollary of Theorems 4 and 2, the session key K_v, exported in the control actions, is equivalent to a variable bound to a fresh, independent name, since φ_4 contains $\nu N.\{K_v = N\}$. Hence, up to observational equivalence, K_v is syntactically independent from $\mathcal{S}$ and the values intercepted by the attacker. As previously discussed [Abadi and Fournet 2001], this provides a characterization of perfect forward secrecy for the session key. We obtain this property even with our liberal reuse of exponentials. We also derive a more specific (but still comforting) property that K_v is distinct from any key established in another session of the protocol.

Independently, ProVerif confirms that the key K_v exchanged between two compliant principals remains secret—that is, here, the adversary cannot compute it—even if we give the long-term secret keys K_-^A of all principals to the attacker after the end of the protocol run. Similarly, ProVerif verifies that all signing keys K_-^A for $A \in \mathcal{C}$ and Diffie-Hellman exponents d_x for $x \in X$ remain secret.

7.3 Identity Protection

We can rely on observational equivalence also for identity protection. The intercepted variables defined by φ_3 and φ_4 are independent from ID_A, ID'_R, and ID_B; this property is a strong privacy guarantee for sessions between compliant principals. Further guarantees can be obtained with particular hypotheses (see Abadi and Fournet [2004]). For instance, if all identities in S_I^B are of the form ID_A for some $A \in \mathcal{C}$ (that is, B does not accept sessions with the attacker) and there is no input on *init*^{B} (that is, B is only a responder) then, using Theorems 4 and 2, we easily check that the identity ID_B occurs only in outputs on *connect*^{A} and otherwise cannot be observed by an active attacker.

Relying on the technique of Blanchet [2004], ProVerif can prove some identity-protection properties stated as observational equivalences.

—To show that the identity of the responder is protected against active attacks, we consider configurations in which two responders can have their signing keys among K_-^A and K_-^B (so their identities among $\mathsf{Pk}(K_-^A)$ and $\mathsf{Pk}(K_-^B)$), and we show that these configurations are observationally equivalent. Hence, an adversary cannot distinguish a configuration in which a responder uses K_-^A from one in which it uses K_-^B, and it cannot tell whether two responders use the same signing key.

—Similarly, to show that identities of compliant principals are protected against passive attacks, we consider configurations in which two responders and two initiators can have their signing keys among K_-^A, K_-^B, K_-^C, K_-^D, and we show that these configurations are observationally equivalent. In these configurations, the attacker can listen but not send messages on channel c.

These configurations contain other responders and initiators, with other keys. The channels *accept*^{A} and *connect*^{A} are restricted, so that an adversary cannot observe messages sent on these channels. All responders accept connections only from compliant principals (only for simplicity). We also prove a similar observational equivalence that shows that JFKi protects the identity of the initiator.

On the other hand, we have also found limitations in identity protection; see Section 2.3.

8. PLAUSIBLE DENIABILITY

Plausible deniability [Roe 1997] is an explicit (and controversial) design goal for IP security. In the context of session establishment, plausible deniability entails limiting the amount of evidence that can be gathered by an active attacker in order to prove the existence and characteristics of past sessions, including sessions with the attacker. An extreme protocol would leak no such evidence during session establishment, leaving the choice of any non-repudiation mechanism to the application layer. (Further discussions of plausible deniability appear, for example, in Internet Drafts [Harkins et al. 2002] and in the work of Mao and Paterson on interactions and trade-offs between deniability and authentication [2003].)

As is the case with privacy properties, plausible deniability depends on any a priori knowledge of the behavior of the principals. For instance, if A is known to use a signing key only as a JFKr responder, and to accept sessions with at most B, then any signature from A proves that A actually accepted a session with B, irrespective of the signature contents. This knowledge can be formalized via contexts that define the behavior of principals under scrutiny, as proposed by Abadi and Fournet [2004]. In our example, the context would represent A's behavior.

In general, a principal A can deny communicating with B if, for any given (data) evidence, there exists an active attacker that could obtain the same evidence by interacting with A although A did not attempt to communicate with B. Accordingly, for a given series of transitions representing the denied actions, we consider any alternative "plausible actions" that may have led to the production of the same evidence.

Therefore, in JFKr, when A is a compliant principal in a configuration $\mathcal{S}$, we may say that the trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$ is plausibly explained by $\mathcal{S}_a \xrightarrow{\eta_a} \mathcal{S}'_a$ when, for some evaluation context C that does not restrict variables defined in $\mathcal{S}'$, we have the static equivalence $C[\mathcal{S}'_a] \approx_s \mathcal{S}'$. (Static equivalence, presented in Section 3.5, is observational equivalence with respect to contexts that can use the terms exported by active substitutions, but cannot otherwise interact with the protocol.) Informally, $\mathcal{S}_a$ represents an alternative to the initial configuration $\mathcal{S}$, and C defines alternative computations for the terms exported by $\mathcal{S}'$. For instance, if $\mathcal{S}'$ exports a message encrypted under a key that can be computed by the context, and $\mathcal{S}'$ does not use that key at all, C may perform a computation to produce that encrypted message. On the other hand, if $\mathcal{S}'$ exports a message signed with the key of a compliant principal, all plausible explanations of $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$ should include a corresponding signature; the signature cannot be blamed on the context.

Since JFKr participants sign only session-specific values, rather than identities, JFKr should offer some plausible deniability properties. (JFKi does not, by design.) The next theorem states some such properties. Its proof, in Appendix C, relies on trace rewriting (much as for Theorem 4).

THEOREM 5 (DENIABILITY IN JFKR). *For any normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$, there is a normal trace $\mathcal{S}_a \xrightarrow{\eta_a} \mathcal{S}'_a$ for any configuration $\mathcal{S}_a$ and actions η_a obtained from $\mathcal{S}$ and η by*

(*1*) *erasing any session between compliant principals (as detailed in Theorem 4);*

(*2*) *removing from $\mathcal{C}$ any A that performs no control actions (as long as $\mathcal{C} \neq \emptyset$);*

(*3*) *in any init^A input, modifying the terms $\mathsf{ID}'_R, \mathsf{sa}_I$ (as long as the trace is normal);*

(*4*) *erasing any connect^A output;*

(*5*) *in any accept^A output, modifying the terms $\mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I$ (as long as the trace is normal) so that the new ID_I is a verification key in S_A^I not equal to ID_B for any $B \in \mathcal{C}$;*

(*6*) *replacing S_I^A with any set of terms that contains all ID_Is in the remaining accept^A outputs.*

and there is an evaluation context C that does not restrict any variable defined in $\mathcal{S}_a$ such that $C[\mathcal{S}'_a] \approx_s \mathcal{S}'$.

Thus, compliant principals can deny sessions among themselves, or even their presence unless they are target of an active attack. Moreover, they can deny the parameters of any session with a dishonest principal (but not the existence or number of such sessions).

We are currently attempting to develop a more general theory of plausible deniability. This theory should be applicable to a broad class of protocols, including JFKr. We expect that, a posteriori, the present results about JFKr will exemplify that theory. (We note that the study of other security properties often proceeds similarly. Informal discussions and specific results often precede general theories. Even afterwards, the analyses of particular protocols often refer to those general theories only loosely.)

9. CONCLUSION

Despite a substantial body of work on the formal analysis of security protocols, and despite much interest in IKE and related protocols, it seems that neither IKE nor its successors has been the subject of an exhaustive analysis until now. The conference paper that presents JFK argues informally about some of its core properties, and calls for a formal analysis; the later journal paper includes some more detailed arguments. Recent work by Datta et al. [2002; 2004; 2005] explores how the STS protocol, two JFK variants, and the core of IKE can be derived by successive refinements. In particular, it discusses the properties of JFKr, and isolates the usage of authenticators for DOS-resistance and of encrypted signatures for identity protection (without however precise claims or proofs). Further afield, the literature contains partial but useful machine-assisted verifications of IKE and Skeme (a protocol that influenced IKE) [Meadows 1999; Blanchet 2001; 2002], and a framework for the study of DOS [Meadows 2001]. More broadly, the literature contains several formal techniques for protocol analysis and many examples, e.g., [Kemmerer et al. 1994; Lowe 1996; Paulson 1998; Thayer Fábrega et al. 1998; Abadi and Gordon 1999; Lincoln et al. 1998; Bodei 2000].

While a number of those techniques could potentially yield at least partial results on JFK, we believe that the use of the applied pi calculus is particularly appropriate. It permits a rich formalization of the protocol; the formulation of some of its properties via process equivalences and others in terms of behaviors; and proofs (sometimes automatic ones) that rely on language-based methods. The effort required seems reasonable enough: our protocol formalization took (roughly) a few months of work, as did the proofs and the ProVerif extensions. While some of the work was difficult and certainly not linear—formalization and proofs required many iterations—it should be easier in future protocol analyses, partly because the use of ProVerif should be more routine. In particular, it took only a few hours to adapt our formalization and re-run the automated proofs for a refinement of JFKr discussed in Section 2.1 and for JFKi.

We regard the present analysis of JFK as an important case study that goes beyond what we have previously attempted, first because JFK is an attractive and intricate "state-of-the-art" protocol of possible practical impact (through its influence on IKEv2 and other protocols), because JFK tightly packages many ideas that appear elsewhere in the field, and also because our analysis explores properties that are central to JFK but that are not often, if ever, explained rigorously. Furthermore, as noted in the introduction, this case study contributes to the development of ideas and results for the specification and verification of security protocols that should be useful beyond the analysis of JFK.

An obvious next problem is the analysis of IKEv2. We have not undertaken it (instead or in addition to the analysis of JFK) because IKEv2 continued to evolve, with influence from JFK and other sources, until relatively recently. (The RFC that describes IKEv2 is from December 2005; our work started in 2002 and was mostly complete well before IKEv2 was stable.) Fortunately, there seems to be substantial awareness of the benefits of formal analysis in and around the IETF, so one may look forward to rigorous studies of IKEv2 and other significant protocols.

Acknowledgments. We are grateful to Ran Canetti and Angelos Keromytis for information on JFK, to John Mitchell for information on his research on refinement, to Véronique Cortier for comments on a draft of this paper, and to Michael Roe and Dieter Gollmann for early discussions on this work. Martín Abadi's work at the University of California was partly supported by the National Science Foundation under Grants CCR-0204162, CCR-0208800, and CCF-0524078. Bruno Blanchet's work was partly done while he was at Max-Planck-Institut für Informatik, Saarbrücken.

REFERENCES

ABADI, M. AND BLANCHET, B. 2005a. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM 52,* 1, 102–146.

ABADI, M. AND BLANCHET, B. 2005b. Computer-assisted verification of a protocol for certified email. *Science of Computer Programming 58,* 1–2 (Oct.), 3–27.

ABADI, M. AND FOURNET, C. 2001. Mobile values, new names, and secure communication. In *28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)*. 104–115.

ABADI, M. AND FOURNET, C. 2004. Private authentication. *Theoretical Computer Science 322,* 3 (Sept.), 427–476. Parts of this work were presented at PET'02 (LNCS 2482) and ISSS'02 (LNCS 2602).

ABADI, M. AND GORDON, A. D. 1999. A calculus for cryptographic protocols: The spi calculus. *Information and Computation 148,* 1 (Jan.), 1–70. An extended version appeared as Digital Equipment Corporation Systems Research Center report No. 149, January 1998.

AIELLO, W., BELLOVIN, S., BLAZE, M., CANETTI, R., IONNIDIS, J., KEROMYTIS, A., AND REINGOLD, O. 2002a. Efficient, DoS-resistant, secure key exchange for internet protocols. In *9th ACM Conference on Computer and Communications Security (CCS'02)*. 48–58.

AIELLO, W., BELLOVIN, S., BLAZE, M., CANETTI, R., IONNIDIS, J., KEROMYTIS, A., AND REINGOLD, O. 2002b. Just fast keying (JFK). IETF Internet Draft `draft-ietf-ipsec-jfk-04.txt`.

AIELLO, W., BELLOVIN, S., BLAZE, M., CANETTI, R., IONNIDIS, J., KEROMYTIS, A., AND REINGOLD, O. 2004. Just fast keying: Key agreement in a hostile internet. *ACM Transactions on Information and System Security 7,* 2 (May), 1–30.

BLANCHET, B. 2001. An efficient cryptographic protocol verifier based on Prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*. 82–96.

BLANCHET, B. 2002. From secrecy to authenticity in security protocols. In *Static Analysis, 9th International Symposium (SAS'02)*. LNCS, vol. 2477. Springer-Verlag, 342–359.

BLANCHET, B. 2004. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy*. 86–100.

BLANCHET, B., ABADI, M., AND FOURNET, C. 2005. Automated verification of selected equivalences for security protocols. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*. IEEE Computer Society, 331–340.

BODEI, C. 2000. Security issues in process calculi. Ph.D. thesis, Università di Pisa.

DATTA, A., DEREK, A., MITCHELL, J. C., AND PAVLOVIC, D. 2004. Abstraction and refinement in protocol derivation. In *17th IEEE Computer Security Foundations Workshop (CSFW-17)*. 30–45.

DATTA, A., DEREK, A., MITCHELL, J. C., AND PAVLOVIC, D. 2005. A derivation system and compositional logic for security protocols. *Journal of Computer Security 13,* 3, 423–482.

DATTA, A., MITCHELL, J. C., AND PAVLOVIC, D. 2002. Derivation of the JFK protocol. `http://www.stanford.edu/~danupam/composition.ps`.

HARKINS, D. AND CARREL, D. 1998. RFC 2409: The Internet Key Exchange (IKE). `http://www.ietf.org/rfc/rfc2409.txt`.

HARKINS, D., KAUFMAN, C., KIVINEN, T., KENT, S., AND PERLMAN, R. 2002. Design rationale for IKEv2. IETF Internet Draft (expired) `draft-ietf-ipsec-ikev2-rationale-00.txt`.

KARN, P. AND SIMPSON, W. 1999. RFC 2522: Photuris: Session-key management protocol. `http://www.ietf.org/rfc/rfc2522.txt`.

KAUFMAN, C. 2005. RFC 4306: Internet Key Exchange (IKEv2) Protocol. `http://www.ietf.org/rfc/rfc4306.txt`.

KEMMERER, R., MEADOWS, C., AND MILLEN, J. 1994. Three systems for cryptographic protocol analysis. *Journal of Cryptology 7,* 2 (Spring), 79–130.

LINCOLN, P., MITCHELL, J., MITCHELL, M., AND SCEDROV, A. 1998. A probabilistic poly-time framework for protocol analysis. In *Fifth ACM Conference on Computer and Communications Security (CCS'98)*. 112–121.

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*. LNCS, vol. 1055. Springer-Verlag, 147–166.

MAO, W. AND PATERSON, K. G. 2003. On the plausible deniability feature of internet protocols. Unpublished manuscript.

MEADOWS, C. 1999. Analysis of the Internet Key Exchange protocol using the NRL protocol analyzer. In *IEEE Symposium on Security and Privacy*. 216–231.

MEADOWS, C. 2001. A cost-based framework for analysis of denial of service networks. *Journal of Computer Security 9,* 1/2, 143–164.

NEEDHAM, R. M. AND SCHROEDER, M. D. 1978. Using encryption for authentication in large networks of computers. *Communications of the ACM 21,* 12 (Dec.), 993–999.

PAULSON, L. C. 1998. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security 6,* 1–2, 85–128.

ROE, M. 1997. Cryptography and evidence. Ph.D. thesis, Clare College, University of Cambrige, UK. Available at `http://research.microsoft.com/users/mroe/thesis.pdf`.

SANGIORGI, D. AND WALKER, D. 2001. *The Pi-calculus: a Theory of Mobile Processes*. Cambridge University Press.

THAYER FÁBREGA, F. J., HERZOG, J. C., AND GUTTMAN, J. D. 1998. Strand spaces: Why is a security protocol correct? In *IEEE Symposium on Security and Privacy*. 160–171.

WAGNER, D. AND SCHNEIER, B. 1996. Analysis of the SSL 3.0 protocol. In *2nd USENIX Workshop on Electronic Commerce*. Usenix Press, 29–40.

Appendix

The appendices give proofs for the results stated in the body of the paper. Appendix A contains the proofs of two lemmas that allow us to eliminate the authenticator cache, thus facilitating other proofs. Appendix B groups the proofs of correspondence assertions, which rely on the same methodology and lemmas. It also includes justifications and details for our use of ProVerif. Finally, Appendix C presents the remaining proofs.

A. PROOFS ON THE TREATMENT OF COOKIES (LEMMAS 1 AND 2)

We give the proofs for the lemmas on DOS resistance stated in Section 6, Lemmas 1 and 2. Our proofs partly rely on standard pi calculus techniques such as bisimulation up to context; see Sangiorgi and Walker [2001], Abadi and Fournet [2001] for additional details.

In order to prove Lemma 1, we first establish that any collections of tokens issued by the receiver and protected by the key K_R are equivalent to fresh, distinct names.

LEMMA 3. *Let $\{K_R\} \uplus \mathcal{N}$ be a finite set of names, and let $(V_{2,N})_{N\in\mathcal{N}}$ be a series of terms where K_R does not occur. Let*

$$\begin{aligned} S^\circ_N &= (\nu t.\{t_N = t\}) \\ S_N &= \{t_N = \mathsf{H}\{K_R\}(N, V_{2,N})\} \end{aligned}$$

We have the static equivalence $\prod_{N\in\mathcal{N}} S^\circ_N \approx_s \nu K_R.\prod_{N\in\mathcal{N}} S_N$.

This lemma is automatically verified by ProVerif, relying on the technique presented by Blanchet et al. [2005]. With these definitions, in a context that binds x_2 to $V_{2,N}$, the processes R°_2 and R_2 of Lemma 1 each have only one transition, which can be written

$$R^\circ_2 \xrightarrow{\nu x_N \overline{c}\langle x_N\rangle} E_2[S^\circ_N \mid R^\circ_3] \qquad \text{and} \qquad R_2 \xrightarrow{\nu x_N \overline{c}\langle x_N\rangle} E_2[S_N]$$

respectively, for the same evaluation context $E_2[\_] = \nu N, t_N.(\{x_N = M_2\} \mid [\_])$.

PROOF OF LEMMA 1. We use the notations R_2°, R_3°, R_2, R_3, and P of Lemma 1 and S_N°, S_N of Lemma 3. We write $R_{3,N}^\circ$ and $R_{3\setminus V}$ instead of R_3° and R_3 to make explicit the free name N and the contents of the cache V (with $V = \emptyset$ initially). After desugaring, we have

$$R_{3,N}^\circ = \nu l. \begin{pmatrix} \overline{l}\langle\rangle \mid !c(z).let\ \{x_3 = \mathsf{F}_4^3(z)\}\ in \\ \qquad if\ z = 3(t_N, N, x_2, x_3)\ then\ l().R_4 \end{pmatrix}$$

$$R_{3\setminus V} = \nu s. \begin{pmatrix} \overline{s}\langle V\rangle \mid !c(z).let\ \{t_N, N, x_2, x_3 = \mathsf{F}_1^3(z), \mathsf{F}_2^3(z), \mathsf{F}_3^3(z), \mathsf{F}_4^3(z)\}\ in \\ \qquad if\ z = 3(t_N, N, x_2, x_3)\ then \\ \qquad if\ t_N = \mathsf{H}\{K_R\}(N, x_2)\ then \\ \qquad s(z').(\overline{s}\langle z'.t_N\rangle \mid if\ z' = z'.t_N\ then\ \mathbf{0}\ else\ R_4) \end{pmatrix}$$

Let $\mathcal{A}$ and $\mathcal{O}$ be finite, disjoint subsets of names (indexing active and dead tokens, respectively) and let

$$\begin{aligned} S^\circ &= \textstyle\prod_{N\in\mathcal{A}\cup\mathcal{O}} S_N^\circ \mid \prod_{N\in\mathcal{A}} R_{3,N}^\circ \mid P \\ S &= \textstyle\prod_{N\in\mathcal{A}\cup\mathcal{O}} S_N \mid R_{3\setminus t_\mathcal{O}} \mid P \end{aligned}$$

where the cache contents $t_\mathcal{O}$ in $R_{3\setminus t_\mathcal{O}}$ is a term that represents the set $\{t_N \mid N \in \mathcal{O}\}$.

We build the relation $\mathcal{R}$ as follows: $\mathcal{R}$ relates all extended processes $S^\circ \mid R^\circ$ and $\nu K_R.(S \mid R)$ where K_R does not occur in $S^\circ \mid R^\circ$ and where R is obtained from R° by substituting $R_2\{x_2 = V_2\}$ for $R_2^\circ\{x_2 = V_2\}$, for some terms V_2 and subprocesses $R_2^\circ\{x_2 = V_2\}$ of R°.

First, we show that $\mathcal{R} \subseteq \approx_s$. We apply Lemma 3 for the set $\{K_R\} \uplus \mathcal{A} \uplus \mathcal{O}$, remark that, by definition, $R^\circ \approx_s R$, and obtain the static equivalence of the frames associated with $S^\circ \mid R^\circ$ and $\nu K_R.(S \mid R)$ by parallel composition. In the rest of the proof, we show that $\mathcal{R} \subseteq \approx$. In the following case analyses for transitions, we rely on $\mathcal{R} \subseteq \approx_s$ to relate the outcome of tests on each side of $\mathcal{R}$.

In S° and S, we say that an input on c *fails* when it is either an input in P or an input followed by a test that will always fail: in $R_{3,N}^\circ$ with $N \in \mathcal{A}$ because matching $3(=t_N, =N, =x_2, x_3)$ fails; in R_3, because matching $3(t_N, N, x_2, x_3)$ or testing $t_N = \mathsf{H}\{K_R\}(N, x_2)$ fails, or because $N \in \mathcal{O}$.

By comparing the definitions of R_3° and R_3, we verify that failing inputs coincide for all messages. If matching or testing in R_3 fails, then matching in $R_{3,N}^\circ$ fails for all $N \in \mathcal{A}$. Conversely, if matching and testing succeed in R_3, then we must have $t_N = t_{N'}$ for some $N' \in \mathcal{A} \uplus \mathcal{O}$. Either $N \in \mathcal{O}$, and the freshness test on t_N fails, or $N \in \mathcal{A}$, the freshness test on t_N succeeds, and matching succeeds in $R_{3,N}^\circ$.

For each $N \in \mathcal{A}$, there may be several inputs on c that do not (necessarily) fail, with a race condition on reading the message $\overline{l}\langle\rangle$ in $R_{3,N}^\circ$ and reading the message $\overline{s}\langle t_\mathcal{O}\rangle$ in R_3, respectively. Accordingly, we let $\mathcal{R}'$ be the smallest relation that contains $\mathcal{R}$ and is closed by matching input transitions on c that may not fail. We represent pairs in $\mathcal{R}'$ as pairs in $\mathcal{R}$ plus sets of inputs indexed by $N \in \mathcal{A}$. We show that $\mathcal{R}'$ is a weak labelled bisimulation up to context and deterministic reduction steps (for failed tests, and for reading the state of the cache).

The condition $\mathcal{R}' \subseteq \approx_s$ follows from $\mathcal{R} \subseteq \approx_s$ and the fact that, in the applied pi calculus, input transitions never affect static equivalence between extended processes. In the case analysis for transitions, we can omit internal communication steps on channel c—those steps can be decomposed into an output followed by an input. This leaves the following cases:

(1) Transitions that affect R° and R are in direct correspondence, except for the transitions of R_2° and R_2 displayed below Lemma 3. For these transitions, we use structural equivalence, add a fresh name N to $\mathcal{A}$, and discard the context $E_2[\_]$ on both sides.

(2) Input transitions on c that fail (on either side) leave the abstract state of the protocol unchanged; they are simulated by an input in P on the other side.

(3) Input transitions on c that may succeed are simulated on both sides using the closure condition of $\mathcal{R}'$, by recording an additional input for some $N \in \mathcal{A}$.

(4) Inputs on a single l or on f, following an input on c that may succeed for the nonce $N \in \mathcal{A}$, lead to the same process R_4 being triggered on both sides. We discard processes resulting from any other non-failed input on c for N. (These inputs now fail.) We discard the replicated input for N in R° (now equivalent to P). We transfer N from $\mathcal{A}$ to $\mathcal{O}$, reflecting the new state of the cache in R_3.

Thus, $\mathcal{R} \subseteq \mathcal{R}' \subseteq \approx$. Finally, the equivalence stated in the lemma is included in $\mathcal{R}$, up to structural equivalence, for $\mathcal{A} = \emptyset$, $\mathcal{O} = \emptyset$, $R^\circ = C[R_2^\circ]$, and $R = C[R_2]$. $\square$

PROOF OF LEMMA 2. This is a corollary of Lemma 1, with auxiliary equivalences to relate our two variants of JFKr configurations to instances of the more abstract statement of Lemma 1.

(1) We insert the replicated input $P = !c(z)$ in parallel with $\mathcal{S}$ and $\mathcal{S}^\circ$. Since $\mathcal{C}$ and X are not empty, we can use the replicated inputs on Messages 1 and 3 to show $\mathcal{S} \mid P \approx \mathcal{S}$ and $\mathcal{S}^\circ \mid P \approx \mathcal{S}^\circ$. Informally, any message on c fails to match (at least) one message pattern $1(\dots)$ or $3(\dots)$, so it can always be received then silently discarded.

(2) We use simple bisimilarities to reorder the fields in Message 3, substituting the tuples (N_I, x_R) and (x_I, e_I, h_I) for the variables x_2 and x_3, and to reorder the tuple of terms in the keyed hash to match the format of Lemma 1.

(3) For each $A \in \mathcal{C}$ in turn, Lemma 1 applied with

$$\begin{aligned} R_4 &= \textstyle\prod_{x \in X} \mathit{if}\ x = x_R\ \mathit{then}\ \mathit{let}\ \kappa_R\ \mathit{in} \dots \qquad \text{(as in } R_3^A) \\ C[\_] &= \textstyle\prod_{x_R \in X} !c(1(N_I, x_I)).[\_] \end{aligned}$$

yields the equivalence $R^A \mid P \approx R^{\star A} \mid P$ where $R^{\star A}$ is

$$\begin{aligned} R^{\star A} = \ & \textstyle\prod_{x_R \in X} \\ & !c(1(N_I, x_I)). \\ & \nu N_R, t_R . \overline{c}\langle 2(N_I, N_R, x_R, \mathrm{g}_R, t_R)\rangle. \\ & ?c(3(=N_I, =N_R, x_I, =x_R, =t_R, e_I, h_I)). \\ & \textstyle\prod_{x \in X} \mathit{if}\ x = x_R\ \mathit{then} \qquad (\star) \\ & \mathit{let}\ \kappa_R\ \mathit{in} \dots \qquad \text{(as in } R_3^A) \end{aligned}$$

that is, $R^{\circ A}$ of Lemma 2 with an additional product at line $(\star)$. In particular, this equivalence holds in the evaluation context that gathers the rest of $\mathcal{S}^\circ \mid P$ and $\mathcal{S} \mid P$ for each A.

(4) Finally, the evaluation context D_X within $\mathcal{S}^\circ$ and $\mathcal{S}$ binds the variables $x_R \in X$ to pairwise-distinct values, so the test in the product of $R^{\star A}$ at line $(\star)$ succeeds exactly

once, for $x = x_R$, and we can replace each $R^{\star A}$ with $R^{\circ A}$ using a simple bisimilarity. $\square$

B. PROOFS OF CORRESPONDENCE PROPERTIES

We give proofs of several correspondence properties: Theorem 1, Theorem 3, and part of Theorem 4. For this purpose we also elaborate on our use of ProVerif, on which the proofs partly rely. Specifically, we discuss events (used for signaling important protocol steps) and correspondence properties; most of this material is not specific to JFK. We also discuss details of our scripts for JFK.

B.1 Events and Normal Traces

In order to keep track of protocol runs precisely using ProVerif, we rely on the insertion of specific actions, named events, that mark important steps of the protocol under study but do not otherwise affect its behavior. In the applied pi calculus, events are just message outputs $\overline{f}\langle M\rangle$ where f is an "event channel" (a name in a particular set $\mathcal{E}$, disjoint from the set of names of ordinary channels). In labelled transitions, output labels for events use "event variables" e. Event variables are not allowed to appear in input labels $a(M)$, so the adversary cannot use them. (This condition is important so that an event $\overline{f}\langle M\rangle$ does not reveal M to the adversary.) Hence, the execution of the process P after inserting events is the execution of P without events, plus the recording of events using labels $\overline{f}\langle e\rangle$ and active substitutions $\{e = M\}$. We extend the conventions on normal traces given in Section 3.6 accordingly:

(1) Event names occur only in outputs; they are neither communicated nor used for inputs in processes and in transitions.
(2) Names and variables extruded in events do not appear in inputs unless they have also been sent on other output channels.

B.2 Correspondence Properties Provable by ProVerif

DEFINITION 1. *A correspondence property p is defined by a series of nested correspondences*

$$p_{\widetilde{l}} = [\mathrm{inj}]\ \alpha_{\widetilde{l}} \rightsquigarrow \bigwedge_{k=1}^{n_{\widetilde{l}}} p_{\widetilde{l}k}$$

indexed by sequences of indices $\widetilde{l} = l_1 \dots l_m$ with $l_i \in [1, \dots, n_{l_1,\dots,l_{i-1}}]$, where [inj] *is an optional* inj *marker, and where $\alpha_{\widetilde{l}}$ is an action.*

The normal trace $P_0 \xrightarrow{\eta} Q$ satisfies property p if and only if there exists a series of partial functions $\chi_{\widetilde{l}}$ on indices of actions in η such that:

(*1*) *for every index ι in η, if the action $\eta(\iota)$ matches α, then*
  (*a*) *$\chi(\iota) = \iota$;*
  (*b*) *there exists a substitution σ such that, for all $\widetilde{l}$, the action $\eta(\chi_{\widetilde{l}}(\iota))$ equals $\alpha_{\widetilde{l}}\sigma$;*
  (*c*) *$\chi_{\widetilde{l}}(\iota) \leq \chi_{\widetilde{m}}(\iota)$ for any $\widetilde{l}$ and any prefix $\widetilde{m}$ of $\widetilde{l}$ (that is, $\eta(\chi_{\widetilde{l}}(\iota))$ occurs before $\eta(\chi_{\widetilde{m}}(\iota))$);*
(*2*) *if $p_{\widetilde{l}}$ has the* inj *marker, then $\chi_{\widetilde{l}}$ is injective.*

The process P_0 satisfies property p if and only if all normal traces $P_0 \xrightarrow{\eta} Q$ of P_0 satisfy p.

This definition of correspondence properties generalizes the ones of Blanchet [2002] in order to capture series of related events in traces. As usual, $\bigwedge$ is conjunction, while $\rightsquigarrow$ and inj are our notations for writing correspondence properties. Intuitively, a correspondence is a tree of actions, and $\widetilde{l}$ is a path in the tree. The index $\widetilde{l}$ represents the nesting of correspondence functions $\chi_{\widetilde{l}}$ that record occurrences of actions $\alpha_{\widetilde{l}}$ in the trace; in particular χ records the occurrences of α. Starting from p, if α appears in the trace, then $\alpha_1, \ldots, \alpha_n$ also occur beforehand. When the inj marker is present before α_i, the correspondence is required to be injective, that is, a distinct α_i must correspond to each α. Moreover, if for example $n_1 > 0$, from p_1 we also have that $\alpha_{11}, \ldots, \alpha_{1n_1}$ appear before α_1. (We rely on compound correspondences instead of multiple, simpler correspondences so that we can index the injective functions from the top-level action α.)

For example, the correspondence property

$$\overline{\mathit{accept}^B}\langle \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle \rightsquigarrow \mathrm{inj}\ \overline{\mathtt{accept}}\langle \mathit{accept}^B, \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$$

means that each output of message $\mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v$ on channel accept^B is preceded by a distinct event $\overline{\mathtt{accept}}\langle \mathit{accept}^B, \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$. More formally, there exists an injective function χ_1 that maps the execution step of the output $\overline{\mathit{accept}^B}\langle \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ to the execution step of the event $\overline{\mathtt{accept}}\langle \mathit{accept}^B, \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ and $\chi_1(\iota) \leq \iota$.

Similarly, using the actions of Theorem 1, the correspondence

$$\overline{\$}\langle N_I\rangle \rightsquigarrow \big(\mathrm{inj}\ c(2(N_I, \_, \_, \_, \_)) \rightsquigarrow \big(\mathrm{inj}\ \overline{c}\langle 1(N_I, \_)\rangle \rightsquigarrow \mathrm{inj}\ \mathit{init}^A(\_, \_)\big)\big)$$

means that each output $\overline{\$}\langle N_I\rangle$ is preceded by actions $\mathit{init}^A(\_, \_)$, $\overline{c}\langle 1(N_I, \_)\rangle$, and $c(2(N_I, \_, \_, \_, \_))$ in that order, and that different outputs $\overline{\$}\langle N_I\rangle$ correspond to different preceding actions. The correspondence

$$\overline{\$}\langle N_I\rangle \rightsquigarrow \mathrm{inj}\ c(2(N_I, \_, \_, \_, \_)) \wedge \mathrm{inj}\ \overline{c}\langle 1(N_I, \_)\rangle \wedge \mathrm{inj}\ \mathit{init}^A(\_, \_)$$

means that each output $\overline{\$}\langle N_I\rangle$ is preceded by actions $\mathit{init}^A(\_, \_)$, $\overline{c}\langle 1(N_I, \_)\rangle$, and $c(2(N_I, \_, \_, \_, \_))$ in any order, and that different outputs $\overline{\$}\langle N_I\rangle$ correspond to different preceding actions.

ProVerif can prove such properties when the actions $\alpha_{\widetilde{l}}$ range over events that include list membership tests $M_{\widetilde{l}} \in M'_{\widetilde{l}}$ in addition to regular events $\overline{f_{\widetilde{l}}}\langle M_{\widetilde{l}}\rangle$. (We say that the actions of a trace match $M \in M'$ simply when M is an element of the list M'.)

B.3 ProVerif Scripts for JFK

The ProVerif scripts that we use for JFKr and JFKi are available at `http://www.di.ens.fr/~blanchet/crypto/jfk.html`. Below we also provide a pi calculus definition of the script for JFKr. This process, written $\mathcal{S}^e$, corresponds to the process $\mathcal{S}$ of Figure 2, except for the differences explained below.

(1) Tagged messages and other tuples rely on primitive constructors and selectors, instead of equations. Similarly, sets rely on primitive operations instead of equations.

(2) Optionally, we eliminate the cache of anti-DOS cookies (in order to get injective correspondences), as detailed in Lemma 2.

(3) A ProVerif script cannot express parallel compositions of subprocesses parameterized by sets, as in $\prod_{A\in\mathcal{C}}$ and $\prod_{x_I\in X}$ (because this defines a family of processes, one for

each set $\mathcal{C}$ and X, instead of a single process). So, instead of a static configuration of compliant principals $A \in \mathcal{C}$, we give an API so that the attacker can allocate and configure compliant principals. Similarly, the script outputs exponents and exponentials on the restricted channel exp and the parallel compositions $\prod_{x_I \in X}$ in I^A and $\prod_{x_R \in X}$ in R^A are replaced with replicated inputs $!exp(d_{x_I}, x_I)$ and $!exp(d_{x_R}, x_R)$, respectively. This is strictly more general, as shown by Lemma 4 below.

We obtain the following definitions:

$$
\begin{aligned}
I^{A'} &= !exp(d_{x_I}, x_I).I_0^A \\
R^{A'} &= \nu K_R.(!exp(d_{x_R}, x_R).R_1^A \mid R_3^{A'}) \\
R_3^{A'} &= !c(3(N_I, N_R, x_I, x_R, t_R, e_I, h_I))\backslash\emptyset. \\
&\quad \mathit{if}\ t_R = \mathsf{H}\{K_R\}(x_R, N_R, N_I)\ \mathit{then\ if}\ t_R\ \mathit{fresh\ then} \\
&\quad ?\,exp(d_{x_R}, =x_R). \\
&\quad \mathit{let}\ \kappa_R\ \mathit{in}\ \ldots \mathit{(as\ in}\ R_3^A\mathit{)} \\
\mathcal{S}^e &= \nu\, exp.\nu\, cp. \\
&\quad (\quad !D_x[\overline{getexp}\langle x\rangle.!\overline{exp}\langle d_x, x\rangle] \\
&\quad \mid\quad !PK^A[\ \nu connect^A, accept^A, init^A, channelS_I^A. \\
&\qquad\qquad \overline{getprinc}\langle \mathsf{ID}_A, init^A, accept^A, connect^A, channelS_I^A\rangle. \\
&\qquad\qquad channelS_I^A(S_I^A). \\
&\qquad\qquad \overline{\mathtt{princ}}\langle K_-^A, \mathsf{ID}_A, init^A, accept^A, connect^A, S_I^A\rangle. \\
&\qquad\qquad (\ !\overline{cp}\langle \mathsf{ID}_A\rangle \mid I^{A'} \mid R^{A'}\ )\quad])
\end{aligned}
$$

The script $\mathcal{S}^e$ uses an event channel $\mathtt{princ}$. Informally, we have one event $\overline{\mathtt{princ}}\langle K_-^A, \mathsf{ID}_A, init^A, accept^A, connect^A, S_I^A\rangle$ for every compliant principal $A \in \mathcal{C}$. In addition, we send all identities ID_A of compliant principals $A \in \mathcal{C}$ on a restricted channel cp. In some proofs, auxiliary processes use these messages to test whether an identity corresponds to a compliant principal of $\mathcal{C}$, by testing equality with any value input from cp.

For any given JFKr configuration $\mathcal{S}$ parameterized by the sets $\mathcal{C}$, X, and $(S_I^A)_{A\in\mathcal{C}}$, we define an "initialization" trace for $\mathcal{S}^e$ that yields a similar configuration, using a series of labels $(\mathcal{C}, X) = (\eta_x)_{x\in X}(\eta'_A)_{A\in\mathcal{C}}(\eta''_A)_{A\in\mathcal{C}}$ where

$$
\begin{aligned}
\eta_x &= \overline{getexp}\langle x\rangle \\
\eta'_A &= \overline{getprinc}\langle \mathsf{ID}_A, init^A, accept^A, connect^A, channelS_I^A\rangle \\
\eta''_A &= channelS_I^A(S_I^A).\overline{\mathtt{princ}}\langle K_-^A, \mathsf{ID}_A, init^A, accept^A, connect^A, S_I^A\rangle
\end{aligned}
$$

and where $(\eta_x)_{x\in X}$ is the concatenation of η_x for all $x \in X$, and similarly for $(\eta'_A)_{A\in\mathcal{C}}$ and $(\eta''_A)_{A\in\mathcal{C}}$. Hence, we allocate all principals $A \in \mathcal{C}$, then provide all their sets S_I^A, in order to enable those sets to include cross-references to compliant principals.

Our next lemma relates the traces of the extended process $\mathcal{S}$ to traces of the script $\mathcal{S}^e$ that include initialization:

LEMMA 4. *For every configuration $\mathcal{S}$ and normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$, there exists a normal trace $\mathcal{S}^e \xrightarrow{(\mathcal{C},X)} Q \xrightarrow{\eta^e} Q'$ with $\mathcal{S}' \approx_s Q'$ such that the actions η^e are those in η interleaved with events, and η^e does not contain* $\mathtt{princ}$ *events.*

An analogous statement holds for $\mathcal{S}^\circ$ instead of $\mathcal{S}$ and for a script without cache instead of $\mathcal{S}^e$, as well as for scripts with additional events in $I^{A'}$ and $R^{A'}$.

PROOF. We detail a process Q obtained after running the trace $\mathcal{S}^e \xrightarrow{(\mathcal{C},X)} Q$.

$$
\begin{array}{rll}
Q & = & \nu exp.\nu cp.D_X[PK^{A_1}[\dots PK^{A_n}[\\
& & (\quad \prod_{x\in X} !\overline{exp}\langle d_x, x\rangle\\
& & \mid\quad \prod_{A\in\mathcal{C}}(\ !\overline{cp}\langle \mathsf{ID}_A\rangle \mid I^{A'} \mid R^{A'}\ )\\
& & \mid\quad !D_x[\overline{getexp}\langle x\rangle.!\overline{exp}\langle d_x, x\rangle]\\
& & \mid\quad !PK^A[\ \nu connect^A, accept^A, init^A, channelS_I^A.\\
& & \qquad\qquad \overline{getprinc}\langle \mathsf{ID}_A, init^A, accept^A, connect^A, channelS_I^A\rangle.\\
& & \qquad\qquad channelS_I^A(S_I^A).\\
& & \qquad\qquad \overline{\mathtt{princ}}\langle K_-^A, \mathsf{ID}_A, init^A, accept^A, connect^A, S_I^A\rangle.\\
& & \qquad\qquad (\ !\overline{cp}\langle \mathsf{ID}_A\rangle \mid I^{A'} \mid R^{A'}\ )\quad])]]]
\end{array}
$$

To simulate the normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$, we rely only on the processes in the first two products, particularly $I^{A'}$ and $R^{A'}$, which closely mirror I^A and R^A within $\mathcal{S}$. The main difference is the introduction of a communication step on restricted channel $getexp$ to dynamically bind d_x, x instead of relying on a static product $\prod_{x\in X}$.

The proof is easily adapted to $\mathcal{S}^\circ$ and other variants of the script (without cache, with additional actions), as these variants do not interfere with the initialization trace. □

B.4 Extending Correspondences from Events to Actions

Lemmas 5 and 8 are not specific to JFK. Lemma 5 can be used to order series of events and actions for a given protocol, depending only on the structure of the protocol. Lemma 8 allows us to infer a correspondence on actions from a correspondence on events. Lemmas 6 and 7 are more specific to JFK, since they depend on the interface for creating compliant principals, but they could also be adapted to other protocols.

The next lemma states that, if a protocol P_0 syntactically contains a series of nested actions $\alpha_1, \dots, \alpha_n$ with no replication between them, then every action α_n that occurs in a trace of P_0 is preceded by a series of actions $\alpha_1, \dots, \alpha_{n-1}$. The last two hypotheses of this lemma guarantee that no other action in P_0 can be an instance of α_n. In this and subsequent lemmas, we associate with each pattern X the open term obtained by replacing all subpatterns of the form $=M$ with M.

LEMMA 5. *Let P_0 be a process of the form $C_1[\alpha_1\sigma.C_2[\alpha_2\sigma \dots C_n[\alpha_n\sigma.P]]]$ where*

—*C_1 is any context; $C_2, \dots, C_n$ are contexts with no replication above the hole;*

—*for all $i \leq n$, either $\alpha_i = \overline{c_i}\langle M_i\rangle$, and we let $\alpha'_i = \alpha_i$; or $\alpha_i = c_i(X_i)$, and we let $\alpha'_i = c_i(M_i)$ where M_i is the term associated with X_i;*

—*for all $j \leq i \leq n$, C_j does not bind the names of α_i;*

—*for all $i < j \leq n$, C_j does not bind the names and variables of $\alpha_i\sigma$;*

—*all output channels in P_0 are names (not variables);*

—*α_n is an output action; c_n does not occur elsewhere as output channel.*

Then P_0 satisfies the correspondence property $\alpha'_n \rightsquigarrow (\mathrm{inj}\ \dots \rightsquigarrow (\mathrm{inj}\ \alpha'_2 \rightsquigarrow \mathrm{inj}\ \alpha'_1))$.

PROOF. We show the following invariant: if $P_0 \xrightarrow{\eta} Q$ is a normal trace, then $Q \equiv Q'$ for some Q' of the form

$$C''\big[\alpha_k\sigma_{jk}.C_{k+1,j,k}[\alpha_{k+1}\sigma_{jk} \ldots C_{njk}[\alpha_n\sigma_{jk}.P_{jk}]]\big]^{k\in\{1,\ldots,n\},j\in\{1,\ldots,l_k\}}$$

where

—C'' is any context (with $l_1 + \cdots + l_n$ holes) that contains no replication above the $l_2 + \cdots + l_n$ holes with indices k, j such that $k \geq 2$; $C_{i'jk}$ is any context with no replication above the hole;

—for all i, i', j, k such that $k < i' \leq i \leq n$ and $j \leq l_k$, C'' and $C_{i'jk}$ do not bind the names of α_i;

—for all i, j, k such that $i < k \leq n$ and $j \leq l_k$, C'' does not bind the names and variables of $\alpha_i\sigma_{jk}$; for all i, i', j, k such that $k < i' \leq n$, $j \leq l_k$, and $i < i'$, $C_{i'jk}$ does not bind the names and variables of $\alpha_i\sigma_{jk}$;

—all output channels in Q' are names (not variables);

—all other occurrences of c_n as output channel in Q' are in the "then" branch of tests that fail (that is, tests of the form *if* $M_1 = M_2$ *then* P where M_1 and M_2 are closed terms and $\Sigma \vdash M_1 \neq M_2$);

—there exist $n-1$ injective functions $(\chi_i)_{i<n}$ that map all indices of actions $\alpha'_n\sigma$ in η to indices of actions $\alpha'_i\sigma$ in η such that, for all ι, $\chi_1(\iota) < \ldots < \chi_{n-1}(\iota) < \iota$; there exist $n-1$ injective functions $(\chi'_i)_{i<n}$ that map all indices j, k such that $i < k \leq n$ and $j \leq l_k$ to indices of actions in η such that $\chi'_i(j,k)$ is the index of an action $\alpha'_i\sigma_{jk}$ in η, $\chi'_1(j,k) < \ldots < \chi'_{k-1}(j,k)$, and the images of all χ_i and χ'_i are pairwise disjoint.

The proof is by induction on the trace $P_0 \xrightarrow{\eta} Q$, starting with $P_0 = Q$ for $l_1 = 1$ and $l_{k>1} = 0$. In the inductive case, for each $k \leq n$, the context C'' has l_k holes filled with processes of the form $\alpha_k\sigma_{jk}.R$.

—When we unfold a replication of C'' above a hole filled with a process of the form $\alpha_1\sigma_{j1}.R$, we increase l_1 by one.

—When we execute an action $\overline{c_k}\langle M\rangle$ by reducing $\alpha_k\sigma_{jk}$ for $k < n$, we decrease l_k by one, increase l_{k+1} by one, and put the context $C_{k+1,j,k}$ in C''. The remaining process is indexed $l_{k+1}, k+1$. We let $\sigma_{l_{k+1},k+1}$ be a substitution equal to σ_{jk} (modulo the equational theory and the frame of Q') such that, after the reduction, C'' does not bind the variables and names of $\alpha_1\sigma_{l_{k+1},k+1}, \ldots, \alpha_k\sigma_{l_{k+1},k+1} = \overline{c_k}\langle M\rangle$. (This is possible because the variables and names of M are not bound by the context C'' obtained after reduction.) The function χ'_k is extended so that $\chi'_k(l_{k+1}, k+1)$ is the index of the action $\overline{c_k}\langle M\rangle = \alpha'_k\sigma_{l_{k+1},k+1}$ in η. For each $i < k$, the domain of χ'_i is reindexed from j, k before the reduction to $l_{k+1}, k+1$ after the reduction.

—When we execute an action $c_k(M)$ by reducing $\alpha_k\sigma_{jk}$ for $k < n$, and M is an instance of $M_k\sigma_{jk}$, we decrease l_k by one, increase l_{k+1} by one, and put the test in the syntactic sugar for $\alpha_k\sigma_{jk}$ and the context $C_{k+1,j,k}$ in C''. The remaining process is indexed $l_{k+1}, k+1$. We let $\sigma_{l_{k+1},k+1}$ be a substitution equal to σ_{jk} (modulo the equational theory and the frame of Q') on the domain of σ_{jk} such that $M_k\sigma_{l_{k+1},k+1} = M$ and C'' does not bind the variables and names of $\alpha_1\sigma_{l_{k+1},k+1}, \ldots, \alpha_k\sigma_{l_{k+1},k+1}$. (This is possible because the variables and names of M are not bound by C''.) The function χ'_k is extended so that $\chi'_k(l_{k+1}, k+1)$ is the index of the action $c_k(M) = \alpha'_k\sigma_{l_{k+1},k+1}$

in η. For each $i < k$, the domain of χ'_i is reindexed from j, k before the reduction to $l_{k+1}, k+1$ after the reduction.

—When we execute an action $c_k(M)$ by reducing $\alpha_k\sigma_{jk}$ for $k < n$, and M is not an instance of $M_k\sigma_{jk}$, we decrease l_k by one, and put the test in the syntactic sugar for $\alpha_k\sigma_{jk}$ and the process that follows $\alpha_k\sigma_{jk}$ in C''. This transformation adds an occurrence of c_n as output channel in C'', in the "then" branch of a test that fails.

—When we execute $\overline{c_n}\langle M\rangle$, we must reduce $\alpha_n\sigma_{jn}$ for some j since all other occurrences of c_n as output channel in Q' are in the "then" branch of tests that fail. In this case, we decrease l_n by one, and put the process P_{jn} in the context C''. For all $i \leq n$, the functions χ_i are extended by $\chi_i(\iota) = \chi'_i(j, n)$ where ι denotes the index of the action $\alpha_n\sigma_{jn}$ in η, and $\chi'_i(j, n)$ becomes undefined after the execution of $\alpha_n\sigma_{jn}$.

—When we execute another input or output transition, the reduced input or output occurs in C'', so the context C'' is modified. In the case of an input transition, variables may be instantiated; this instantiation preserves the invariant.

—When we execute an internal communication, the communication channel is not free in Q' by definition of normal traces, so the reduced input and output are not $\alpha_k\sigma_{jk}$, so they occur in C''. We can then apply the same reasoning as in the previous case.

—When we execute a conditional, we decrease l_k for each context in the branch of the conditional that is discarded by the reduction.

The lemma follows from the existence of $\chi_1, \ldots, \chi_{n-1}$ for all $P_0 \xrightarrow{\eta} Q$. $\square$

The next lemma treats events that record the unfolding of compliant principals $A \in \mathcal{C}$.

LEMMA 6. *Let Q be a process of the form*

$$Q \equiv \nu cp.C\big[\prod_{A\in\mathcal{C}} !\overline{cp}\langle \mathsf{ID}_A\rangle \\ \mid \prod_{A\in\mathcal{C}} C'_A[\overline{c_A}\langle \mathsf{ID}'_A, M_A\rangle.(P_A \mid cp(=\mathsf{ID}'_A).\overline{f}\langle c_A, \mathsf{ID}'_A, M_A\rangle)]\big]$$

where

—*C is an evaluation context; C'_A is any context for each $A \in \mathcal{C}$;*

—*f is an event channel; f and cp do not occur anywhere else;*

—*all output channels are names;*

—*for all $A \in \mathcal{C}$, c_A does not occur anywhere else as output channel, and C and C'_A do not bind c_A.*

Assume that we have a normal trace $Q \xrightarrow{\eta^e} Q'$, with no internal communication step on channel cp. Then there is an extended normal trace

$$Q \xrightarrow{\eta^e} Q' \xrightarrow{\overline{f}\langle c_1, \mathsf{ID}'_1, M_1\rangle \ldots \overline{f}\langle c_n, \mathsf{ID}'_n, M_n\rangle} Q''$$

and an injective function χ that maps all indices of actions in η^e matching $\overline{c_A}\langle \mathsf{ID}_B, M\rangle$ with $B \in \mathcal{C}$ to indices of the additional actions $\overline{f}\langle c_A, \mathsf{ID}_B, M\rangle$.

Relying on correspondence properties of the form $\overline{f}\langle c, \mathsf{ID}', M\rangle \leadsto \ldots$ for the extended normal trace, we can thus establish properties of the form "for each action $\overline{c}\langle \mathsf{ID}_B, M\rangle$ such that $B \in \mathcal{C}, \ldots$" for the trace $Q \xrightarrow{\eta^e} Q'$.

PROOF. With a proof similar to that of Lemma 5, we show the following invariant:

$$\begin{aligned} Q' \equiv \nu cp.C_0\big[& \textstyle\prod_{A\in\mathcal{C}} !\overline{cp}\langle \mathsf{ID}_A\rangle \\ & \mid C'\left[\overline{c_i}\langle \mathsf{ID}'_i, M_i\rangle.(P_i \mid cp(=\mathsf{ID}'_i).\overline{f}\langle c_i, \mathsf{ID}'_i, M_i\rangle)\right]^{1\le i\le k} \\ & \mid \textstyle\prod_{k<i\le k+l} cp(=\mathsf{ID}'_i).\overline{f}\langle c_i, \mathsf{ID}'_i, M_i\rangle\big] \end{aligned}$$

where

—C_0 is an evaluation context; C' is a context with k holes;

—f and cp do not occur anywhere else;

—all output channels are names;

—$c_i = c_A$ for some $A \in \mathcal{C}$, for each $i = 1 \dots k + l$; for all $A \in \mathcal{C}$, c_A does not occur anywhere else as output channel, and C_0 and C' do not bind c_A;

—there exists an injective function χ'' mapping indices of actions $\overline{c_A}\langle \mathsf{ID}', M\rangle$ in η to indices i such that $k < i \le k + l$, $c_A = c_i$, $\mathsf{ID}' = \mathsf{ID}'_i$, and $M = M_i$.

In Q', the k processes within context C' keep track of all instances of outputs on c_A for all $A \in \mathcal{C}$; the l processes in the final product keep track of all inputs on cp in evaluation context (released after communications on c_A in C').

Using the invariant, we complete the proof as follows. For each $\overline{c_A}\langle \mathsf{ID}_B, M\rangle$ that occurs in η at step ι, $\chi''(\iota)$ is an index i such that $k < i \le k + l$, $c_A = c_i$, $\mathsf{ID}_B = \mathsf{ID}'_i$, and $M = M_i$, so the process $cp(=\mathsf{ID}_B).\overline{f}\langle c_A, \mathsf{ID}_B, M\rangle$ occurs in Q'. If $B \in \mathcal{C}$, then $!\overline{cp}\langle \mathsf{ID}_B\rangle$ also occurs in Q' and we reduce Q' into Q'' by executing

$$\begin{aligned} !\overline{cp}\langle \mathsf{ID}_B\rangle \mid cp(=\mathsf{ID}_B).\overline{f}\langle c_A, \mathsf{ID}_B, M\rangle \quad & \to\to \quad !\overline{cp}\langle \mathsf{ID}_B\rangle \mid \overline{f}\langle c_A, \mathsf{ID}_B, M\rangle \\ & \xrightarrow{\nu e.\overline{f}\langle e\rangle} !\overline{cp}\langle \mathsf{ID}_B\rangle \mid \{e = (c_A, \mathsf{ID}_B, M)\} \end{aligned}$$

in some evaluation context. We let χ map ι to the index of the action $\overline{f}\langle c_A, \mathsf{ID}_B, M\rangle$. □

The next lemma exploits the structure of the interface for creating compliant principals and the `princ` events to infer a correspondence property with fixed compliant principals $A_1, \dots, A_n$ from a correspondence property in which compliant principals are represented by variables.

LEMMA 7. *Suppose that* $\mathcal{S}^e \xrightarrow{(\mathcal{C},X)} Q$ *and that* $\mathcal{S}^e$ *satisfies the correspondence property* $\alpha \rightsquigarrow p_1 \wedge \bigwedge_{i=1}^{n} \overline{\mathtt{princ}}\langle K^i_-, \mathsf{ID}_i, \mathit{init}^i, \mathit{accept}^i, \mathit{connect}^i, S^i_I\rangle$ *where* α *and* p_1 *contain only events but no* `princ` *events,* α *contains no function symbol with ProVerif equations, and, for each* $i \le n$*, either* K^i_- *or* ID_i *or* init^i *or* accept^i *or* $\mathit{connect}^i$ *occurs in* α*. Let* Q' *be* Q *after replacing the* `princ` *events and their guarded processes with* $\mathbf{0}$*. For each* $\widetilde{A} = (A_1, \dots, A_n) \in \mathcal{C}^n$*, the process* Q' *satisfies the correspondence property* $\alpha\sigma_{\widetilde{A}} \rightsquigarrow p_1\sigma_{\widetilde{A}}$*, where* $\sigma_{\widetilde{A}} = \prod_{i=1}^{n}(\{K^i_- = K^{A_i}_-\} \mid \{\mathsf{ID}_i = \mathsf{ID}_{A_i}\} \mid \{\mathit{init}^i = \mathit{init}^{A_i}\} \mid \{\mathit{accept}^i = \mathit{accept}^{A_i}\} \mid \{\mathit{connect}^i = \mathit{connect}^{A_i}\} \mid \{S^i_I = S^{A_i}_I\})$.

PROOF. Assume that p_1 is of the form $p_{\widetilde{l}} = [\mathrm{inj}]\ \alpha_{\widetilde{l}} \rightsquigarrow \bigwedge_{k=1}^{n_{\widetilde{l}}} p_{\widetilde{l}k}$. Let $Q' \xrightarrow{\eta}$ be a normal trace of Q'. Then $Q \xrightarrow{\eta}$ is also a normal trace of Q without `princ` events, hence $\mathcal{S}^e \xrightarrow{(\mathcal{C},X)} Q \xrightarrow{\eta}$ is a normal trace of $\mathcal{S}^e$, so it satisfies the correspondence property $\alpha \rightsquigarrow p_1 \wedge \bigwedge_{i=1}^{n} \overline{\mathtt{princ}}\langle K^i_-, \mathsf{ID}_i, \mathit{init}^i, \mathit{accept}^i, \mathit{connect}^i, S^i_I\rangle$. Thus, since $(\mathcal{C}, X)$ contains only `princ` events and η contains no `princ` events, there exists a series of partial functions $\chi_{\widetilde{l}}$ on indices of actions in η such that:

(1) for every index ι in η, if the action $\eta(\iota)$ matches $\alpha\sigma_{\widetilde{A}}$, then it matches α, so
   (a) $\chi(\iota) = \iota$;
   (b) there exists a substitution σ' such that, for all $\widetilde{l}$ empty or beginning with 1, the action $\eta(\chi_{\widetilde{l}}(\iota))$ equals $\alpha_{\widetilde{l}}\sigma'$ and, for all $i \leq n$, the action $(\mathcal{C}, X)(\chi_{i+1}(\iota))$ equals $\overline{\mathtt{princ}}\langle K^i_-, \mathsf{ID}_i, \mathit{init}^i, \mathit{accept}^i, \mathit{connect}^i, S^i_I\rangle\sigma'$; so by the form of $\mathtt{princ}$ events in $(\mathcal{C}, X)$, there exists $\widetilde{A}' = (A'_1, \dots, A'_n)$ such that $\overline{\mathtt{princ}}\langle K^i_-, \mathsf{ID}_i, \mathit{init}^i, \mathit{accept}^i, \mathit{connect}^i, S^i_I\rangle\sigma' = \overline{\mathtt{princ}}\langle K^i_-, \mathsf{ID}_i, \mathit{init}^i, \mathit{accept}^i, \mathit{connect}^i, S^i_I\rangle\sigma_{\widetilde{A}'}$, hence there exists σ'' such that $\sigma' = \sigma_{\widetilde{A}'}\sigma''$; so, $\alpha\sigma_{\widetilde{A}'}\sigma''$ matches $\alpha\sigma_{\widetilde{A}}$; so $A_i = A'_i$ for all $i \leq n$, since for each $i \leq n$, K^i_- or ID_i or init^i or accept^i or $\mathit{connect}^i$ occurs in α; so for all $\widetilde{l}$ empty or beginning with 1, the action $\eta(\chi_{\widetilde{l}}(\iota))$ equals $\alpha_{\widetilde{l}}\sigma' = \alpha_{\widetilde{l}}\sigma_{\widetilde{A}}\sigma''$;
   (c) $\chi_{\widetilde{l}}(\iota) \leq \chi_{\widetilde{m}}(\iota)$ for any $\widetilde{l}$ and any prefix $\widetilde{m}$ of $\widetilde{l}$;
(2) if $p_{\widetilde{l}}$ has the inj marker, then $\chi_{\widetilde{l}}$ is injective.

Therefore, by definition, Q' satisfies the correspondence property $\alpha\sigma_{\widetilde{A}} \rightsquigarrow p_1\sigma_{\widetilde{A}}$. □

The next lemma allows us to infer a correspondence on input and output actions from a correspondence on events proved by ProVerif. The root action of the inferred correspondence must still be the same event as in the correspondence proved by ProVerif; we use Lemma 5 or Lemma 6 to change the root action.

LEMMA 8. *Let P_0 be a process. Consider a partial function ϕ from events to actions defined by a finite number of cases of one of the three following forms:*

—$\phi(\alpha) = \alpha$.
—$\phi(\alpha) = \alpha' = c(M)$*, such that all variables of α' also occur in α, α contains no function symbol with ProVerif equations, and either* $P_0 = C''\big[c(X\sigma_j).C_j[\alpha\sigma_j.P_j]\big]^{j\in\{1,\dots,l\}}$ *or* $P_0 = C''\big[?c(X\sigma_j).C_j[\alpha\sigma_j.P_j]\big]^{j\in\{1,\dots,l\}}$ *where M is the term associated with the pattern X; C'' is a context that does not bind the names of α and α'; each C_j is a context that consists of any number of tests above the hole; and no other event in P_0 unifies with α.*
—$\phi(\alpha) = \alpha' = \overline{c}\langle M\rangle$*, such that all variables of α' also occur in α, α contains no function symbol with ProVerif equations, and* $P_0 = C''\big[\alpha\sigma_j.\alpha'\sigma_j.P_j\big]^{j\in\{1,\dots,l\}}$ *where C'' does not bind the names of α and α' and no other event in P_0 unifies with α.*

We assume that, when two cases $\phi(\alpha_1) = \alpha'_1$ and $\phi(\alpha_2) = \alpha'_2$ occur in the definition of ϕ, α_1 and α_2 have different event channels. We extend ϕ by $\phi(\alpha\sigma) = \alpha'\sigma$ if $\phi(\alpha) = \alpha'$, and extend ϕ to correspondences by applying ϕ to each action in the correspondence.

Let p be a correspondence with events in the domain of ϕ and such that $\phi(\alpha_0) = \alpha_0$ for the root action α_0 of p. If P_0 satisfies p, then P_0 also satisfies $\phi(p)$.

PROOF. We consider a subset $\mathcal{T}$ of the normal traces of P_0 such that

—if α is an event such that $\phi(\alpha)$ is an input action, and α is executed, then the input action $\phi(\alpha)$ occurs in the trace just before α (with only internal actions in between);
—if α is an event such that $\phi(\alpha)$ is an output action, and α is executed, then the output action $\phi(\alpha)$ occurs in the trace just after α (with only internal actions in between).

Since P_0 satisfies p, all traces of P_0 in $\mathcal{T}$ satisfy p and, by construction of $\mathcal{T}$, these traces also satisfy $\phi(p)$. Let us now consider any normal trace $P_0 \xrightarrow{\eta} Q$ of P_0. By commuting internal actions and events with other actions, we build a normal trace $P_0 \xrightarrow{\eta'} Q'$ in $\mathcal{T}$ such that the actions in the image of ϕ occur in the same order in η' as in η.

For each case $\phi(\alpha) = \alpha'$ of the definition of ϕ with $\alpha' \neq \alpha$, starting from a trace $P_0 \xrightarrow{\eta} Q$, we build a trace $P_0 \xrightarrow{\eta''} Q''$ such that η'' is equal to η except that instances of α may have been moved or deleted, and every instance of α in the resulting trace satisfies the conditions for being in $\mathcal{T}$. The construction distinguishes input cases and output cases:

—Case $\phi(\alpha) = \alpha' = c(M)$, such that all variables of α' also occur in α, α contains no function symbol with ProVerif equations, and $P_0 = C''\big[c(X\sigma_j).C_j[\alpha\sigma_j.P_j]\big]^{j\in\{1,\dots,l\}}$ where M is the term associated with the pattern X; C'' is a context that does not bind the names of α and α'; each C_j is a context that consists of any number of tests above the hole; and no other event in P_0 unifies with α. (The case $?c(X\sigma_j)$ instead of $c(X\sigma_j)$ can be handled in a similar way, though the syntactic sugar for $?c(X\sigma_j)$ requires keeping track of additional reductions after the input.) For each action $\alpha'' = \alpha\sigma''$ that occurs in η, we show that $\phi(\alpha'')$ is executed before α'' in η, and we build a trace $P_0 \xrightarrow{\eta''} Q$ by executing internal actions followed by α'' just after $\phi(\alpha'')$. To this end, we establish the following invariant by induction on the length of the trace $P_0 \xrightarrow{\eta} Q$ (as in Lemma 5): if $P_0 \xrightarrow{\eta} Q$, then

$$Q \equiv C''\big[c(X\sigma_j).C_j[\alpha\sigma_j.P_j]\big]^{j\in\{1,\dots,l\}}\big[C'_j[\alpha\sigma'_j.P'_j]\big]^{j\in\{1,\dots,l'\}}$$

where

—C'' is a context (with $l + l'$ holes) that does not bind the names of α and α', with only restrictions and parallel compositions above the l' holes filled with $C'_j[\alpha\sigma'_j.P'_j]$;

—C'' does not bind the variables and names of $c(X\sigma'_j)$ for any $j \le l'$;

—C_j for all $j \le l$ and C'_j for all $j \le l'$ are contexts that consist of any number of tests above the hole;

—all other events in Q that unify with α are in the "then" branch of tests that fail;

—there exists a trace $P_0 \xrightarrow{\eta''} Q$ such that η'' is equal to η except that some instances of α have been moved so that the actions $\alpha\sigma'$ for any σ' are immediately preceded by $\alpha'\sigma'$; there exists an injective function χ that maps each index $j \le l'$ to the index of an action $c(M\sigma'_j)$ in the trace $P_0 \xrightarrow{\eta''} Q$ not immediately followed by an instance of α in η''; and there exists an injective function χ' that maps each index $j \le l'$ to the index of the transition that puts $C'_j[\alpha\sigma'_j.P'_j]$ in evaluation context, such that $\chi(j) \le \chi'(j)$ and all steps between $\chi(j)$ (excluded) and $\chi'(j)$ (included) are silent reduction steps.

For any step that does not involve any process in the l' last holes of C'', we carry over the same step to the η'' trace. In particular, when an input $c(X\sigma_j)$ in one of the first l holes is reduced, we increment l', let C'_j be an instance of C_j guarded by the pattern-matching test introduced by the syntactic sugar for the pattern X, and set $\chi(l')$ and $\chi'(l')$ to the current index.

For $j \le l'$, if a test in C'_j succeeds, we update C'_j by removing this test; if a test in C'_j fails, we decrement l'; if C'_j is empty and the event $\alpha\sigma'_j$ is executed, we decrement l'. For any step of one of these three forms, the step commutes with all preceding steps in the η'' trace, up to the step $\chi'(j)$ that puts $C'_j[\alpha\sigma'_j.P'_j]$ in evaluation context; we move the new step there, and increment $\chi'(j)$ if this step is a test that succeeds.

—Case $\phi(\alpha) = \alpha' = \overline{c}\langle M\rangle$ such that all variables of α' also occur in α, α contains no function symbol with ProVerif equations, and $P_0 = C''\big[\alpha\sigma_j.\alpha'\sigma_j.P_j\big]^{j\in\{1,\dots,l\}}$ where C'' does not bind the names of α and α' and no other event in P_0 unifies with α.

We build a trace $P_0 \xrightarrow{\eta''} Q''$ by delaying the execution of all actions $\alpha'' = \alpha\sigma''$ that occur in η until just before the execution of the corresponding $\phi(\alpha'')$, if that corresponding $\phi(\alpha'')$ is executed in η. Otherwise, α'' is not executed in $P_0 \xrightarrow{\eta''} Q''$. To this end, we establish the following invariant by induction on the length of the trace $P_0 \xrightarrow{\eta} Q$: if $P_0 \xrightarrow{\eta} Q$, then

$$Q \equiv C''\big[\alpha\sigma_j.\alpha'\sigma_j.P_j\big]^{j\in\{1,\dots,l\}}\big[\alpha'\sigma'_j.P'_j\big]^{j\in\{1,\dots,l'\}}$$

where no other event in P_0 unifies with α; C'' is a context (with $l + l'$ holes) that does not bind the names of α and α', with only restrictions and parallel compositions above the l' holes filled with $\alpha'\sigma'_j.P'_j$; there exists a trace $P_0 \xrightarrow{\eta''} Q''$ such that $Q'' \equiv C''\big[\alpha\sigma_j.\alpha'\sigma_j.P_j\big]^{j\in\{1,\dots,l\}}\big[\alpha\sigma'_j.\alpha'\sigma'_j.P'_j\big]^{j\in\{1,\dots,l'\}}$ where η'' is equal to η except that some instances of α have been moved or deleted so that the actions $\alpha\sigma'$ for any σ' are immediately followed by $\alpha'\sigma'$.

(When $\alpha\sigma_j$ is reduced in $P_0 \xrightarrow{\eta} Q$, we do not execute that reduction in $P_0 \xrightarrow{\eta''} Q''$. When $\alpha'\sigma'_j$ is reduced in $P_0 \xrightarrow{\eta} Q$, we reduce $\alpha\sigma'_j$ and $\alpha'\sigma'_j$ in $P_0 \xrightarrow{\eta''} Q''$. For all other reductions, we execute the same reduction in $P_0 \xrightarrow{\eta} Q$ and $P_0 \xrightarrow{\eta''} Q''$.)

After applying this construction for all cases, the resulting trace $P_0 \xrightarrow{\eta'} Q'$ is in $\mathcal{T}$, so $P_0 \xrightarrow{\eta'} Q'$ satisfies $\phi(p)$. The actions in the image of ϕ occur in the same order in η and η', so $P_0 \xrightarrow{\eta} Q$ satisfies $\phi(p)$, and thus P_0 satisfies $\phi(p)$. □

B.5 Proofs for JFK (Theorems 1, 3, and part of 4)

We detail the proofs of Theorems 1, 3, and the first point of Theorem 4.

PROOF OF THEOREM 1. Theorem 1 is a direct consequence of Lemma 5, in combination with Lemma 2 to remove the cache. For the first part of Theorem 1, Lemma 5 applied to $\mathcal{S}_\$$ yields the correspondence

$$\overline{\$}\langle N_I\rangle \rightsquigarrow \big(\text{inj } c(2(N_I,\_,\_,\_,\_)) \rightsquigarrow \big(\text{inj } \overline{c}\langle 1(N_I,\_)\rangle \rightsquigarrow \text{inj } \mathit{init}^A(\_,\_)\big)\big)$$

For the second part of Theorem 1, let $\mathcal{S}^\circ_\$$ be $\mathcal{S}^\circ$ with an additional output $\overline{\$}\langle N_I, N_R\rangle$ before the Diffie-Hellman computation of κ_R in $R^{\circ A}$. Lemma 5 applied to $\mathcal{S}^\circ_\$$ yields the correspondence

$$\begin{aligned}\overline{\$}\langle N_I, N_R\rangle \rightsquigarrow \big(&\text{inj } c(3(N_I, N_R,\_,\_,\_,\_,\_)) \rightsquigarrow \\ &\big(\text{inj } \overline{c}\langle 2(N_I, N_R,\_,\_,\_)\rangle \rightsquigarrow \text{inj } c(1(N_I,\_))\big)\big)\end{aligned}$$

A variant of Lemma 2 (with the additional \$ outputs) shows that $\mathcal{S}^\circ_\$$ and $\mathcal{S}_\$$ have the same normal traces, so the same correspondence holds for $\mathcal{S}_\$$. □

PROOF OF THEOREM 3. The proof of this theorem and of the first point of Theorem 4 uses the following technique. Since ProVerif cannot prove correspondences directly on input and output actions, we instrument the ProVerif script with events after each relevant input and before each relevant output. We use ProVerif to show that the obtained script satisfies a correspondence property p on events. Next, we consider a trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$ of the system $\mathcal{S}$. By Lemma 4, we obtain a corresponding trace $\mathcal{S}^e \xrightarrow{(\mathcal{C},X)} Q \xrightarrow{\eta^e} \mathcal{S}''$ of the ProVerif script. We show by Lemma 7 that the process Q', obtained by removing the $\mathtt{princ}$ events and processes under them in Q, satisfies an adapted correspondence p'. By Lemma 8, we infer that Q' also satisfies a correspondence $\phi(p')$ in which some events have been replaced with input or output actions (if needed). By Lemma 5 or 6, the root event of the correspondence $\phi(p')$ is executed, so, by the correspondence $\phi(p')$, we conclude that the desired actions have been executed in the trace.

(1) The proof of the first part of Property 1 relies on a script $\mathcal{S}^e$ instrumented with an event $\overline{\mathtt{accept}}\langle \mathit{accept}^A, \mathsf{ID}_I, \mathsf{ID}_{R'}, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ just before $\overline{\mathit{accept}}^A\langle \mathsf{ID}_I, \mathsf{ID}_{R'}, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ in R_3^A. ProVerif shows that $\mathcal{S}^e$ satisfies the correspondence property

$$\begin{aligned}&\overline{\mathtt{accept}}\langle \mathit{accept}, \mathsf{ID}_1, \mathsf{ID}_{R'}, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle \rightsquigarrow\\ &\quad\overline{\mathtt{princ}}\langle K_-, \mathsf{ID}, \mathit{init}, \mathit{accept}, \mathit{connect}, S_I\rangle \wedge (\mathsf{ID}_1 \in S_I)\end{aligned} \tag{1}$$

For some given $\mathcal{C}$, X, and $(S_I^A)_{A\in\mathcal{C}}$, assume that we have a normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$. By Lemma 4, we also have a normal trace $\mathcal{S}^e \xrightarrow{(\mathcal{C},X)} Q \xrightarrow{\eta^e} \mathcal{S}''$ with $\mathcal{S}'' \approx_s \mathcal{S}'$ where the actions η^e are those in η interleaved with events and η^e does not contain $\mathtt{princ}$ events. Let Q' be Q after replacing the $\mathtt{princ}$ events and their guarded processes with $\mathbf{0}$. By the correspondence property (1) and Lemma 7, Q' satisfies the correspondence property

$$\overline{\mathtt{accept}}\langle \mathit{accept}^B, \mathsf{ID}_1, \mathsf{ID}_{R'}, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle \rightsquigarrow \mathsf{ID}_1 \in S_I^B \tag{2}$$

For any $B \in \mathcal{C}$, if $\overline{\mathit{accept}}^B\langle \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ appears in η, it also occurs in $(\mathcal{C}, X)\eta^e$. We apply Lemma 5 to the process Q. Since all output channels of Q are names and accept^B is not used anywhere else as a channel, Q satisfies

$$\overline{\mathit{accept}}^B\langle \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle \rightsquigarrow \text{inj } \overline{\mathtt{accept}}\langle \mathit{accept}^B, \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$$

Hence $\overline{\mathtt{accept}}\langle \mathit{accept}^B, \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ occurs in η^e, which is also a trace of Q'. So, by the correspondence property (2), $\mathsf{ID}_A \in S_I^B$.

(2) The proof of the second part of Property 1 relies on a script $\mathcal{S}^e$, without cache and with an extra event $\overline{\mathtt{init}}\langle \mathit{init}^A, \mathsf{ID}'_R, \mathsf{sa}_I\rangle$ just after the input init^A in I_0^A and an extra process $cp(= \mathsf{ID}_I).\overline{\mathtt{accepthonest}}\langle \mathit{accept}^A, \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ just after the output on accept^A in R_3^A, in parallel with the continuation that sends Message 4. ProVerif shows that $\mathcal{S}^e$ satisfies the correspondence property

$$\begin{aligned}&\overline{\mathtt{accepthonest}}\langle \mathit{accept}^2, \mathsf{ID}_1, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle \rightsquigarrow\\ &\qquad \text{inj } \overline{\mathtt{init}}\langle \mathit{init}^1, \mathsf{ID}'_R, \mathsf{sa}_I\rangle\\ &\quad \wedge\ \overline{\mathtt{princ}}\langle K_-^1, \mathsf{ID}_1, \mathit{init}^1, \mathit{accept}^1, \mathit{connect}^1, S_I^1\rangle\\ &\quad \wedge\ \overline{\mathtt{princ}}\langle K_-^2, \mathsf{ID}_2, \mathit{init}^2, \mathit{accept}^2, \mathit{connect}^2, S_I^2\rangle\end{aligned} \tag{3}$$

For some given $\mathcal{C}$, X, and $(S_I^A)_{A\in\mathcal{C}}$, assume that we have a normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$. By Lemma 2, we have a normal trace $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}'^\circ$. By Lemma 4, we also have a normal trace $\mathcal{S}^e \xrightarrow{(\mathcal{C},X)} Q \xrightarrow{\eta^e} \mathcal{S}''$ with no internal communications on channel cp and with

$\mathcal{S}'' \approx_s \mathcal{S}'^{\circ}$. The actions η^e are those in η interleaved with events, and η^e does not contain `princ` events.

Let Q' be Q after replacing the `princ` events and their guarded processes with $\mathbf{0}$. By the correspondence property (3) and Lemma 7, Q' satisfies the correspondence property

$$\overline{\texttt{accepthonest}}\langle \mathit{accept}^B, \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle \rightsquigarrow \text{inj } \overline{\texttt{init}}\langle \mathit{init}^A, \mathsf{ID}'_R, \mathsf{sa}_I\rangle$$

By Lemma 8 applied with $\phi(\overline{\texttt{init}}\langle \mathit{init}^A, \mathsf{ID}'_R, \mathsf{sa}_I\rangle) = \mathit{init}^A(\mathsf{ID}'_R, \mathsf{sa}_I)$ and ϕ equal to the identity on `accepthonest` events, Q' satisfies the correspondence property

$$\overline{\texttt{accepthonest}}\langle \mathit{accept}^B, \mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle \rightsquigarrow \text{inj } \mathit{init}^A(\mathsf{ID}'_R, \mathsf{sa}_I) \tag{4}$$

The actions η^e also label a trace of Q', $Q' \xrightarrow{\eta^e}$. By Lemma 6 applied to Q', the trace $Q' \xrightarrow{\eta^e}$ can be extended with an event $\overline{\texttt{accepthonest}}\langle \mathit{accept}^B, \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ for each $\overline{\mathit{accept}^B}\langle \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ in $(\mathcal{C}, X)\eta^e$ with $\mathsf{ID}_I = \mathsf{ID}_A$ for some $A \in \mathcal{C}$. (The lemma is applied with $c_A = \mathit{accept}^A$ and $f = $ `accepthonest`.) By the correspondence property (4), a distinct preceding input $\mathit{init}^A(\mathsf{ID}'_R, \mathsf{sa}_I)$ occurs in the extended trace, so in η^e since the extended trace adds only `connecthonest` events, so in η.

(3) The proof of Property 2 is done similarly, by adding the events $\overline{\texttt{init}}\langle \mathit{init}^A, \mathsf{ID}'_R, \mathsf{sa}_I\rangle$ just after the init message and $\overline{\texttt{connect}}\langle \mathit{connect}^A, \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ just before the connect message, for the first part, and by adding $\overline{\texttt{accept}}\langle \mathit{accept}^A, \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ just before the accept message and $\mathit{cp}(= \mathsf{ID}_R).\overline{\texttt{connecthonest}}\langle \mathit{connect}^A, \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ just after the connect message, for the second part. $\square$

PROOF OF FIRST POINT OF THEOREM 4. The proof is done by considering a script $\mathcal{S}^e$ without cache and with additional events after inputs and just before outputs:

—$\overline{\texttt{init}}\langle \mathit{init}^A, \mathsf{ID}'_R, \mathsf{sa}_I\rangle$ (added just after the input init^A) records that I receives the init message $\mathsf{ID}'_R, \mathsf{sa}_I$ on channel init^A.

—$\overline{\texttt{mess1}}\langle \mathsf{ID}_A, N_I, x_I, \mathit{init}^A, \mathsf{ID}'_R, \mathsf{sa}_I\rangle$ (added just before sending Message 1) records that I sends the message $1(N_I, x_I)$.

—$\overline{\texttt{mess1rec}}\langle \mathsf{ID}_A, N_I, x_I\rangle$ (added just after receiving Message 1) records that R receives the message $1(N_I, x_I)$.

—$\overline{\texttt{mess2}}\langle \mathsf{ID}_A, N_I, N_R, x_I, x_R, \mathsf{g}_R, t_R\rangle$ (added just before sending Message 2) records that R sends the message $2(N_I, N_R, x_R, \mathsf{g}_R, t_R)$.

—$\overline{\texttt{mess2rec}}\langle \mathsf{ID}_A, N_I, N_R, x_R, \mathsf{g}_R, t_R, x_I, \mathsf{ID}'_R, \mathsf{sa}_I\rangle$ (added just after receiving Message 2) records that I receives the message $2(N_I, N_R, x_R, \mathsf{g}_R, t_R)$.

—$\overline{\texttt{mess3}}\langle \mathsf{ID}_A, N_I, N_R, x_I, x_R, t_R, e_I, h_I, \mathsf{g}_R, \mathsf{ID}'_R, \mathsf{sa}_I, K_v\rangle$ (added just before sending message 3) records that I sends the message $3(N_I, N_R, x_I, x_R, t_R, e_I, h_I)$.

—$\overline{\texttt{mess3rec}}\langle \mathsf{ID}_A, N_I, N_R, x_I, x_R, t_R, e_I, h_I\rangle$ (added just after $?c(3(= N_I, = N_R, x_I, = x_R, = t_R, e_I, h_I)))$ records that R receives the message $3(N_I, N_R, x_I, x_R, t_R, e_I, h_I)$.

—$\overline{\texttt{accept}}\langle \mathit{accept}^A, \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ (added just before sending the accept message) records that R executes $\overline{\mathit{accept}^A}\langle \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$.

—$\overline{\texttt{mess4}}\langle \mathsf{ID}_A, \mathsf{ID}_I, e_R, h_R, N_I, N_R, x_I, x_R, t_R, e_I, h_I\rangle$ (added just before sending Message 4) records that R sends the message $4(e_R, h_R)$.

—$\overline{\texttt{mess4rec}}\langle \mathsf{ID}_A, e_R, h_R, \mathit{connect}^A, \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ (added after receiving, decrypting, and verifying Message 4) records that I received the message $4(e_R, h_R)$.

—$cp(= \mathsf{ID}_R).\overline{\texttt{connecthonest}}\langle \mathit{connect}^A, \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ is added just after the connect message. The event records that I executes $\overline{\mathit{connect}}^A\langle \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ with $A \in \mathcal{C}$ and $\mathsf{ID}_R = \mathsf{ID}_B$ for some $B \in \mathcal{C}$.

We sometimes mention in events more variables than those that occur in the corresponding input or output. These additional variables allow us to store in the event more information on the state of the principal, and help ProVerif perform its proof.

Let us define the following actions:

$$\begin{aligned}
\alpha_{\texttt{connecthonest}} &= \overline{\texttt{connecthonest}}\langle \mathit{connect}^1, \mathsf{ID}_2, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle\\
\alpha_{\texttt{princ1}} &= \overline{\texttt{princ}}\langle _, \mathsf{ID}_1, \mathit{init}^1, \mathit{accept}^1, \mathit{connect}^1\rangle\\
\alpha_{\texttt{princ2}} &= \overline{\texttt{princ}}\langle _, \mathsf{ID}_2, \mathit{init}^2, \mathit{accept}^2, \mathit{connect}^2\rangle\\
\alpha_{\overline{\mathit{connect}}} &= \overline{\mathit{connect}^1}\langle \mathsf{ID}_2, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle\\
\alpha_{\texttt{mess4rec}} &= \overline{\texttt{mess4rec}}\langle \mathsf{ID}_1, e_R, h_R, \mathit{connect}^1, \mathsf{ID}_2, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle\\
\alpha_{c(4)} &= c(4(e_R, h_R))\\
\alpha_{\overline{c}\langle 4\rangle} &= \overline{c}\langle 4(e_R, h_R)\rangle\\
\alpha_{\texttt{mess4}} &= \overline{\texttt{mess4}}\langle \mathsf{ID}_2, \mathsf{ID}_1, e_R, h_R, N_I, N_R, x_I, x_R, t_R, e_I, h_I\rangle\\
\alpha_{\overline{\mathit{accept}}} &= \overline{\mathit{accept}^2}\langle \mathsf{ID}_1, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle\\
\alpha_{\texttt{accept}} &= \overline{\texttt{accept}}\langle \mathit{accept}^2, \mathsf{ID}_1, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle\\
\alpha_{\texttt{mess3rec}} &= \overline{\texttt{mess3rec}}\langle \mathsf{ID}_2, N_I, N_R, x_I, x_R, t_R, e_I, h_I\rangle\\
\alpha_{c(3)} &= c(3(N_I, N_R, x_I, x_R, t_R, e_I, h_I))\\
\alpha_{\overline{c}\langle 3\rangle} &= \overline{c}\langle 3(N_I, N_R, x_I, x_R, t'_R, e_I, h_I)\rangle\\
\alpha_{\texttt{mess3}} &= \overline{\texttt{mess3}}\langle \mathsf{ID}_1, N_I, N_R, x_I, x_R, t'_R, e_I, h_I, \mathsf{g}_R, \mathsf{ID}'_R, \mathsf{sa}_I, K_v\rangle\\
\alpha_{\texttt{mess2rec}} &= \overline{\texttt{mess2rec}}\langle \mathsf{ID}_1, N_I, N_R, x_R, \mathsf{g}_R, t'_R, x_I, \mathsf{ID}'_R, \mathsf{sa}_I\rangle\\
\alpha_{c(2)} &= c(2(N_I, N_R, x_R, \mathsf{g}_R, t'_R))\\
\alpha_{\overline{c}\langle 2\rangle} &= \overline{c}\langle 2(N_I, N_R, x_R, \mathsf{g}_R, t_R)\rangle\\
\alpha_{\texttt{mess2}} &= \overline{\texttt{mess2}}\langle \mathsf{ID}_2, N_I, N_R, x'_I, x_R, \mathsf{g}_R, t_R\rangle\\
\alpha_{\texttt{mess1rec}} &= \overline{\texttt{mess1rec}}\langle \mathsf{ID}_2, N_I, x'_I\rangle\\
\alpha_{c(1)} &= c(1(N_I, x'_I))\\
\alpha_{\overline{c}\langle 1\rangle} &= \overline{c}\langle 1(N_I, x_I)\rangle\\
\alpha_{\texttt{mess1}} &= \overline{\texttt{mess1}}\langle \mathsf{ID}_1, N_I, x_I, \mathit{init}^1, \mathsf{ID}'_R, \mathsf{sa}_I\rangle\\
\alpha_{\texttt{init}} &= \overline{\texttt{init}}\langle \mathit{init}^1, \mathsf{ID}'_R, \mathsf{sa}_I\rangle\\
\alpha_{\mathit{init}} &= \mathit{init}^1(\mathsf{ID}'_R, \mathsf{sa}_I)
\end{aligned}$$

ProVerif shows that $\mathcal{S}^e$ satisfies the following correspondence property:

$$\begin{aligned}
&\alpha_{\texttt{connecthonest}} \rightsquigarrow \alpha_{\texttt{princ2}} \wedge \alpha_{\texttt{princ1}} \wedge (\text{inj } \alpha_{\texttt{mess4rec}} \rightsquigarrow (\text{inj } \alpha_{\texttt{mess4}} \rightsquigarrow\\
&\quad (\text{inj } \alpha_{\texttt{accept}} \rightsquigarrow (\text{inj } \alpha_{\texttt{mess3rec}} \rightsquigarrow (\text{inj } \alpha_{\texttt{mess3}} \rightsquigarrow (\text{inj } \alpha_{\texttt{mess2rec}} \rightsquigarrow\\
&\quad (\text{inj } \alpha_{\texttt{mess2}} \rightsquigarrow \text{inj } \alpha_{\texttt{mess1rec}}) \wedge (\text{inj } \alpha_{\texttt{mess1}} \rightsquigarrow \text{inj } \alpha_{\texttt{init}})))))))
\end{aligned} \tag{5}$$

For some given $\mathcal{C}$, X, and $(S_I^A)_{A\in\mathcal{C}}$, assume that we have a normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$. By Lemma 2, we have a normal trace $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}'^\circ$. By Lemma 4, we also have a normal trace $\mathcal{S}^e \xrightarrow{(\mathcal{C},X)} Q \xrightarrow{\eta^e} \mathcal{S}''$ with no internal communications on cp and with $\mathcal{S}'' \approx_s \mathcal{S}'^\circ$, where the actions η^e are those in η interleaved with events and η^e does not contain `princ` events.

Let Q' be Q after replacing the `princ` events and their guarded processes with $\mathbf{0}$. By the correspondence property (5) and Lemma 7, Q' satisfies the correspondence property

$$\begin{aligned}&\alpha_{\texttt{connecthonest}}\sigma \rightsquigarrow (\text{inj } \alpha_{\texttt{mess4rec}}\sigma \rightsquigarrow (\text{inj } \alpha_{\texttt{mess4}}\sigma \rightsquigarrow\\ &\quad (\text{inj } \alpha_{\texttt{accept}}\sigma \rightsquigarrow (\text{inj } \alpha_{\texttt{mess3rec}}\sigma \rightsquigarrow (\text{inj } \alpha_{\texttt{mess3}}\sigma \rightsquigarrow (\text{inj } \alpha_{\texttt{mess2rec}}\sigma \rightsquigarrow\\ &\quad (\text{inj } \alpha_{\texttt{mess2}}\sigma \rightsquigarrow \text{inj } \alpha_{\texttt{mess1rec}}\sigma) \wedge (\text{inj } \alpha_{\texttt{mess1}}\sigma \rightsquigarrow \text{inj } \alpha_{\texttt{init}}\sigma)))))))\end{aligned}$$

where $\sigma = \{K_-^1 = K_-^A\} \mid \{\mathsf{ID}_1 = \mathsf{ID}_A\} \mid \{\mathit{init}^1 = \mathit{init}^A\} \mid \{\mathit{accept}^1 = \mathit{accept}^A\} \mid \{\mathit{connect}^1 = \mathit{connect}^A\} \mid \{S_I^1 = S_I^A\} \mid \{K_-^2 = K_-^B\} \mid \{\mathsf{ID}_2 = \mathsf{ID}_B\} \mid \{\mathit{init}^2 = \mathit{init}^B\} \mid \{\mathit{accept}^2 = \mathit{accept}^B\} \mid \{\mathit{connect}^2 = \mathit{connect}^B\} \mid \{S_I^2 = S_I^B\}$. We apply Lemma 8 with $\phi(\alpha_{\texttt{mess4rec}}\sigma) = \alpha_{c(4)}\sigma$, $\phi(\alpha_{\texttt{mess4}}\sigma) = \alpha_{\overline{c}\langle 4\rangle}\sigma$, $\phi(\alpha_{\texttt{accept}}\sigma) = \alpha_{\overline{\mathit{accept}}}\sigma$, $\phi(\alpha_{\texttt{mess3rec}}\sigma) = \alpha_{c(3)}\sigma$, $\phi(\alpha_{\texttt{mess3}}\sigma) = \alpha_{\overline{c}\langle 3\rangle}\sigma$, $\phi(\alpha_{\texttt{mess2rec}}\sigma) = \alpha_{c(2)}\sigma$, $\phi(\alpha_{\texttt{mess2}}\sigma) = \alpha_{\overline{c}\langle 2\rangle}\sigma$, $\phi(\alpha_{\texttt{mess1rec}}\sigma) = \alpha_{c(1)}\sigma$, $\phi(\alpha_{\texttt{mess1}}\sigma) = \alpha_{\overline{c}\langle 1\rangle}\sigma$, $\phi(\alpha_{\texttt{init}}\sigma) = \alpha_{\mathit{init}}\sigma$, and ϕ equal to the identity on `connecthonest` events. Then Q' satisfies the correspondence property

$$\begin{aligned}&\alpha_{\texttt{connecthonest}}\sigma \rightsquigarrow (\text{inj } \alpha_{c(4)}\sigma \rightsquigarrow (\text{inj } \alpha_{\overline{c}\langle 4\rangle}\sigma \rightsquigarrow\\ &\quad (\text{inj } \alpha_{\overline{\mathit{accept}}}\sigma \rightsquigarrow (\text{inj } \alpha_{c(3)}\sigma \rightsquigarrow (\text{inj } \alpha_{\overline{c}\langle 3\rangle}\sigma \rightsquigarrow (\text{inj } \alpha_{c(2)}\sigma \rightsquigarrow\\ &\quad (\text{inj } \alpha_{\overline{c}\langle 2\rangle}\sigma \rightsquigarrow \text{inj } \alpha_{c(1)}\sigma) \wedge (\text{inj } \alpha_{\overline{c}\langle 1\rangle}\sigma \rightsquigarrow \text{inj } \alpha_{\mathit{init}}\sigma)))))))\end{aligned}$$

ProVerif cannot show automatically that Message 1 is received by R after it is sent by I, but a simple manual argument shows it: the nonce N_I is created just before sending Message 1, and appears in the Message 1 received by R, so it must indeed have been received after I has sent it. So Q' satisfies the correspondence property

$$\begin{aligned}&\alpha_{\texttt{connecthonest}}\sigma \rightsquigarrow (\text{inj } \alpha_{c(4)}\sigma \rightsquigarrow (\text{inj } \alpha_{\overline{c}\langle 4\rangle}\sigma \rightsquigarrow\\ &\quad (\text{inj } \alpha_{\overline{\mathit{accept}}}\sigma \rightsquigarrow (\text{inj } \alpha_{c(3)}\sigma \rightsquigarrow (\text{inj } \alpha_{\overline{c}\langle 3\rangle}\sigma \rightsquigarrow (\text{inj } \alpha_{c(2)}\sigma \rightsquigarrow\\ &\quad (\text{inj } \alpha_{\overline{c}\langle 2\rangle}\sigma \rightsquigarrow (\text{inj } \alpha_{c(1)}\sigma \rightsquigarrow (\text{inj } \alpha_{\overline{c}\langle 1\rangle}\sigma \rightsquigarrow \text{inj } \alpha_{\mathit{init}}\sigma)))))))))\end{aligned} \tag{6}$$

The actions η^e also label a trace of Q', $Q' \xrightarrow{\eta^e}$. By Lemma 6 applied to Q', the trace $Q' \xrightarrow{\eta^e}$ can be extended with an event $\overline{\texttt{connecthonest}}\langle \mathit{connect}^A, \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ for each $\overline{\mathit{connect}}^A\langle \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ with $A \in \mathcal{C}$ and $\mathsf{ID}_R = \mathsf{ID}_B$ for some $B \in \mathcal{C}$. (The lemma is applied with $c_A = \mathit{connect}^A$ and $f = \texttt{connecthonest}$.)

In the extended trace, for each $\overline{\mathit{connect}}^A\langle \mathsf{ID}_B, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$ with $A \in \mathcal{C}$ and $B \in \mathcal{C}$, we have $\overline{\texttt{connecthonest}}\langle \mathit{connect}^A, \mathsf{ID}_B, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle$. Then the correspondence property (6) shows that the actions required by Theorem 4 have been executed in order in the extended trace, therefore also in η^e since the extended trace adds only `connecthonest` events, and therefore in η, since η^e adds only events to η. $\square$

C. OTHER PROOFS OF SECRECY, AUTHENTICITY, AND DENIABILITY

Finally, we give proofs of the remaining results of Section 7 and the theorem of Section 8. These rely on an analysis of configurations.

$$
\begin{array}{rcl}
I_0 & = & !\mathit{init}^A(\mathsf{ID}'_R, \mathsf{sa}_I).I_1 \\
I_1 & = & \nu N_I.(\varphi_I \mid \overline{c}\langle \mathsf{1}(N_I, x_I)\rangle.I_2) \\
I_2 & = & c(\mathsf{2}(=N_I, N_R, x_R, \mathsf{g}_R, t_R)).I_3 \\
I_3 & = & \nu K_a, K_e, K_v, s_I, e_I, h_I.(\kappa_I \mid \sigma_I \mid \overline{c}\langle \mathsf{3}(N_I, N_R, x_I, x_R, t_R, e_I, h_I)\rangle.I_4) \\
I_4 & = & c(\mathsf{4}(e_R, h_R)).\mathit{if}\ \mathsf{H}\{K_a\}(\mathsf{r}, e_R) = h_R\ \mathit{then}\ I_5 \\
I_5 & = & \nu \mathsf{ID}_R, \mathsf{sa}_R, s_R.(\tau_I \mid \mathit{if}\ \mathsf{V}\{\mathsf{ID}_R\}(s_R, (N_I, N_R, x_I, x_R))\ \mathit{then}\ I_{5a}) \\
I_{5a} & = & \overline{\mathit{connect}}^A\langle \mathsf{ID}_R, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle \\
I'_5 & = & \varphi_I \mid \nu s_I, s_R, K_a, K_e.(\kappa_I \mid \sigma_I \mid \tau_I) \\
\varphi_I & = & \nu N.\{N_I = N\} \\
\kappa_I & = & \prod_{u=a,e,v}\{K_u = \mathsf{H}\{x_R\,\hat{}\,d_{x_I}\}(N_I, N_R, \mathsf{u})\} \\
\sigma_I & = & \{s_I = \mathsf{S}\{K_-^A\}(N_I, N_R, x_I, x_R, \mathsf{g}_R)\} \mid \\
 & & \{e_I = \mathsf{E}\{K_e\}(\mathsf{ID}_A, \mathsf{ID}'_R, \mathsf{sa}_I, s_I)\} \mid \\
 & & \{h_I = \mathsf{H}\{K_a\}(\mathsf{i}, e_I)\} \\
\tau_I & = & \{\mathsf{ID}_R, \mathsf{sa}_R, s_R = \mathsf{D}\{K_e\}(e_R)\}
\end{array}
$$

$$
\begin{array}{rcl}
R_1 & = & !c(\mathsf{1}(N_I, x_I)).R_2 \\
R_2 & = & \nu N_R, t_R.\big(\varphi_R \mid \overline{c}\langle \mathsf{2}(N_I, N_R, x_R, \mathsf{g}_R, t_R)\rangle.R_3[\overline{l}\langle\rangle]\big) \\
R_3[\_] & = & \nu l.([\_] \mid !c(\mathsf{3}(=N_I, =N_R, x_I, =x_R, =t_R, e_I, h_I)).l().R_{3a}) \\
R_{3a} & = & \nu K_a, K_e, K_v.(\kappa_R \mid \mathit{if}\ \mathsf{H}\{K_a\}(\mathsf{i}, e_I) = h_I\ \mathit{then}\ R''_{3b}) \\
R_{3b} & = & \nu \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, s_I. \\
 & & (\tau_R \mid \mathit{if}\ \mathsf{ID}_I \in S_I^A\ \mathit{then}\ \mathit{if}\ \mathsf{V}\{\mathsf{ID}_I\}(s_I, (N_I, N_R, x_I, x_R, \mathsf{g}_R))\ \mathit{then}\ R_{3c}) \\
R_{3c} & = & \overline{\mathit{accept}}^A\langle \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, \mathsf{sa}_R, K_v\rangle.R_4 \\
R_4 & = & \nu s_R, e_R, h_R.(\sigma_R \mid \overline{c}\langle \mathsf{4}(e_R, h_R)\rangle) \\
R'_4 & = & \varphi_R \mid \nu s_I, s_R, K_a, K_e.(\kappa_R \mid \tau_R \mid \sigma_R) \\
\varphi_R & = & \nu N.\{N_R = N\} \mid \nu N.\{t_R = N\} \\
\kappa_R & = & \prod_{u=a,e,v}\{K_u = \mathsf{H}\{x_I\,\hat{}\,d_{x_R}\}(N_I, N_R, \mathsf{u})\} \\
\sigma_R & = & \{s_R = \mathsf{S}\{K_-^A\}(N_I, N_R, x_I, x_R)\} \mid \\
 & & \{e_R = \mathsf{E}\{K_e\}(\mathsf{ID}_A, \mathsf{sa}_R, s_R)\} \mid \\
 & & \{h_R = \mathsf{H}\{K_a\}(\mathsf{r}, e_R)\} \\
\tau_R & = & \{\mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, s_I = \mathsf{D}\{K_e\}(e_I)\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{S}[\_] & = & D_X\left[PK^{\mathcal{C}}\left[\prod_{A\in\mathcal{C}}(I^A \mid R^A) \mid [\_])\right]\right] \\
I^A & = & \prod_{x_I\in X} I_0 \\
R^A & = & \prod_{x_R\in X} R_1 \\
PK^{\mathcal{C}}[\_] & = & \nu(K_-^A)_{A\in\mathcal{C}}.(\prod_{A\in\mathcal{C}}\{\mathsf{ID}_A = \mathsf{Pk}(K_-^A)\} \mid [\_]) \\
D_x[\_] & = & \nu d_x.(\{x = \mathsf{g}\,\hat{}\,d_x\} \mid [\_]) \\
D_X[\_] & = & D_{x_1}[\ldots D_{x_n}[\_]]\text{ where }X = \{x_1, \ldots, x_n\}
\end{array}
$$

Fig. 3. Linearized JFKr protocol, with auxiliary notations for the session state

C.1 Running Configurations: Definition

We set up notations to decompose arbitrary traces $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}'$ into parallel sessions with an explicit representation of their respective states. (The configuration $\mathcal{S}^\circ$ is the sequential variant of JFKr, obtained by Lemma 2.) Figure 3 restates the definition of the protocol without the responder cache, with auxiliary definitions for various sub-processes. (For convenience, some active substitutions appear in guarded contexts, always under a restriction on their domain; they stand for ordinary substitutions applied within the scope of the restriction.) With these notations, $\mathcal{S}^\circ \equiv \mathcal{S}[\mathbf{0}]$.

Since both roles of the protocol are now sequential, we index the state for each session in each role using a series of action labels that represent the messages processed and generated so far. In a few cases, these labels do not entirely determine the internal state of the session (for instance the internal choice of an exponential $x_I \in X$ when receiving an *init*^{A} message); in those cases, we annotate the trace with that state (for instance writing $(\!(x_I)\!)$ after the *init*^{A} action). In the state, we also do not keep track of unimportant internal reduction steps, such as tests that will always fail or succeed in their given evaluation contexts.

We first describe abstract configurations obtained by interleaving the resulting extended traces, with no formal correspondence with $\mathcal{S}'$ at this stage. Then we study the equational properties of these abstract configurations, as a prerequisite to the equivalences of Theorem 2 and to the case analysis on transitions $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}'$, which depends on equality tests on terms. In particular, we show how to simplify the final state of sessions between compliant principals, which involve exponentials in X. By induction on η, we then show that every normal trace $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}'$ is an interleaving of extended traces that match η, leading to $\mathcal{S}'' \approx \mathcal{S}'$ (up to structural equivalence and deterministic internal steps).

In the following definition, the cases are numbered according to the messages to which they pertain. For example, for the initiator, case (1) corresponds to the session state just before Message 1; case ($\times$3) corresponds to a session that fails before Message 3, after a bad Message 2. For the responder, we distinguish cases (3), (3,1), (3,2), ... The case $(3, n)$ corresponds to a responder that has received n instances of Message 3, and has not selected one of them yet. The case (3a) corresponds to a responder just after the selection of one of those instances. We identify (3,0) and (3).

DEFINITION 2 (EXTENDED TRACES). *For the initiator, extended traces range over:*

(1) $\xrightarrow{\mathit{init}^A(\mathsf{ID}'_R,\mathsf{sa}_I)(\!(x_I)\!)}$ *for any* $A \in \mathcal{C}$, $x_I \in X$, *and terms* ID'_R, sa_I, *with state* I_1.

(2) $\xrightarrow{\mathit{init}^A(\mathsf{ID}'_R,\mathsf{sa}_I)(\!(x_I)\!)\nu N_I.\overline{c}\langle 1(N_I,x_I)\rangle}$ *(which we abbreviate* $\xrightarrow{\eta_2}$*), with state* $\varphi_I \mid I_2$.

(3) $\xrightarrow{\eta_2}\xrightarrow{c(2(=N_I,N_R,x_R,\mathsf{g}_R,t_R))}$ *for any terms* N_R, x_R, g_R, t_R, *with state* $\varphi_I \mid I_3$.

($\times$3) $\xrightarrow{\eta_2}\xrightarrow{c(M_2)}$ *with state* φ_I, *for any other message* M_2.

(4) $\xrightarrow{\eta_2}\xrightarrow{c(2(=N_I,N_R,x_R,\mathsf{g}_R,t_R))\nu e_I,h_I.\overline{c}\langle 3(N_I,N_R,x_I,x_R,t_R,e_I,h_I)\rangle}$ *(which we abbreviate* $\xrightarrow{\eta_4}$*), with state* $\varphi_I \mid \nu K_a, K_e, K_v, s_I.(\kappa_I \mid \sigma_I \mid I_4)$.

(5) $\xrightarrow{\eta_4}\xrightarrow{c(4(e_R,h_R))}$ *for any terms* e_R *and* h_R *such that the two tests in* I_4 *and* I_5 *succeed, with state* $\varphi_I \mid \nu K_a, K_e, K_v, s_I, \mathsf{ID}_R, \mathsf{sa}_R, s_R.(\kappa_I \mid \sigma_I \mid \tau_I \mid I_{5a})$.

($\times$5) $\xrightarrow{\eta_4}\xrightarrow{c(M_4)}$ *for any other message* M_4, *with state* $\varphi_I \mid \nu K_a, K_e, K_v, s_I.(\kappa_I \mid \sigma_I)$.

(5′) $\xrightarrow{\eta_4} \xrightarrow{c(4(e_R,h_R))\nu \mathsf{ID}_R,\mathsf{sa}_R,K_v.\overline{\mathit{connect}}^A\langle \mathsf{ID}_R,\mathsf{ID}'_R,\mathsf{sa}_I,\mathsf{sa}_R,K_v\rangle}$ *with state* I'_5.

For the responder, extended traces range over:

(2) $\xrightarrow{c(1(N_I,x_I))((B,x_R))}$ *for any* $B \in \mathcal{C}$, $x_R \in X$, *and terms* N_I, x_I, *with state* R_2.

(3) $\xrightarrow{c(1(N_I,x_I))((B,x_R))\nu N_R,t_R.\overline{c}\langle 2(N_I,N_R,x_R,\mathsf{g}_R,t_R)\rangle}$ *(which we abbreviate* $\xrightarrow{\eta_3}$*), with state* $\varphi_R \mid R_3[\bar{l}\langle\rangle]$.

(3, n) $\xrightarrow{\eta_3}\xrightarrow{c(M_{3,1})\ldots c(M_{3,n})}$ *for any terms* $M_{3,i}$ *for* $i = 0..n$, *with state* $\varphi_R \mid R_3[\bar{l}\langle\rangle \mid \prod_i l().R_{3a,i}]$ *with a product of instances of* R_{3a}, *which we write* $R_{3a,i}$, *for each* i *such that the message* M_{3i} *matches* $3(=N_I, =N_R, x_I, =x_R, =t_R, e_I, h_I)$.

(3a) $\xrightarrow{\eta_3}\xrightarrow{c(M_{3,1})\ldots c(M_{3,n})((m))}$ *for any* $1 \leq m \leq n$ *such that* (⋆) $M_{3,m}$ *matches* $3(=N_I, =N_R, x_I, =x_R, =t_R, e_I, h_I)$ *for some subterms* x_I, e_I, h_I; *and* (⋆⋆) *the three tests in the resulting process* R_{3a} *succeed, with state* $\varphi_R \mid R_{3a} \mid R'_3$.
($R'_3 = R_3[\prod_{i\neq m} l().R_{3a,i}]$ *is a replicated input on* c *in parallel with deadlocked processes. In its context,* R'_3 *is inert, and could be discarded up to equivalence.)*

(×3a) as above, except for (⋆⋆), with state $\varphi_R \mid R'_3$ *after failing a test.*

(4) $\xrightarrow{\eta_3}\xrightarrow{c(M_{3,1})\ldots c(M_{3,n})((m))\nu \mathsf{ID}_I,\mathsf{ID}'_R,\mathsf{sa}_I,K_v.\overline{\mathit{accept}}^B\langle \mathsf{ID}_I,\mathsf{ID}'_R,\mathsf{sa}_I,\mathsf{sa}_R,K_v\rangle}$ *(which we abbreviate* $\xrightarrow{\eta_4}$*), with state* $\varphi_R \mid \nu s_I, K_a, K_e.(\kappa_R \mid \tau_R \mid R_4 \mid R'_3)$.

(4′) $\xrightarrow{\eta_4}\xrightarrow{\nu e_R,h_R.\overline{c}\langle 4(e_R,h_R)\rangle}$, *with state* $R'_4 \mid R'_3$.

For discarding input messages that do not match a pattern listed above, extended traces finally include inputs $\xrightarrow{c(M)}$ *and* $\xrightarrow{\mathit{init}^A(M)}$ *for any message* M *and* $A \in \mathcal{C}$, *with no state.*

To any interleaving η *of the extended traces listed above, involving any principals of* $\mathcal{C}$ *in any role (with distinct session indices for the variables exported by different sessions), we associate the* abstract configuration $\mathcal{S}[Q]$, *where* Q *is the parallel composition of all session states.*

We first consider the equational net effect of abstract configurations, by analyzing their frames. Let $\varphi(\_)$ be the function on processes that erases any plain process, keeping only restrictions and active substitutions. We compute the frame

$$\varphi(\mathcal{S}[Q]) \;=\; D_X\left[PK^{\mathcal{C}}\left[\textstyle\prod_{A\in\mathcal{C}} \varphi(Q)\right]\right]$$

where $\varphi(Q)$ is a parallel composition of active substitutions given by Definition 2 (with distinct session indices for each session).

For each case of an initiator extended trace, the frame is:

(1) $\mathbf{0}$
(2), (3), (×3) φ_I which exports N_I
(4), (5), (×5) $\varphi_I \mid \nu K_a, K_e, K_v, s_I.(\kappa_I \mid \sigma_I)$ which exports N_I, e_I, h_I
(5′) I'_5 which exports $N_I, e_I, h_I, \mathsf{ID}_R, \mathsf{sa}_R, K_v$

For each case of a responder extended trace, the frame is:

(2) $\mathbf{0}$
$(3, n)$, (3a), (×3a) φ_R which exports N_R, t_R
(4) $\varphi_R \mid \nu s_I, K_a, K_e.(\kappa_R \mid \tau_R)$ which exports $N_R, t_R, \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, K_v$
(4′) R'_4 which exports $N_R, t_R, \mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, K_v, e_R, h_R$

As a special case, consider an abstract configuration $\mathcal{S}'$ with two extended traces, (5′) with initiator A and $\mathsf{ID}_R = \mathsf{ID}_B$ and (4′) with responder B and $\mathsf{ID}_I = \mathsf{ID}_A$, with matching outputs and inputs on channel c. Intuitively, these two extended traces form a successful run of the protocol between A and B. Using equational simplifications (notably the Diffie-Hellman equation), the frames for the two extended traces are $I'_5 \mid R'_4 \equiv \varphi \mid \varphi'$ where

$$\varphi = \varphi_I \mid \varphi_R \mid \nu s_I, s_R, K_a, K_e.(\kappa_I \mid \sigma_I \mid \sigma_R)$$

defines the exchanged messages and $\varphi' = \{K_v^B = K_v\} \mid \{\mathsf{ID}_I^B = \mathsf{ID}_A\} \mid \{\mathsf{ID}_R^A = \mathsf{ID}_B\} \mid \{\mathsf{ID}'^B_R = \mathsf{ID}'_R\} \mid \{\mathsf{sa}_I^B = \mathsf{sa}_I\} \mid \{\mathsf{sa}_R^A = \mathsf{sa}_R\}$ accounts for duplicate variable definitions.

Our next lemma relates an abstract configuration that contains the outcome of these two extended traces (φ) to an abstract configuration that defines instead fresh, distinct names (φ_4 of Section 7.1) using observational equivalence:

LEMMA 9. *If $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}[Q]$ is an interleaving of extended traces, then*

$$\mathcal{S}[Q \mid \varphi] \approx \varphi_4 \mid \mathcal{S}[Q]$$

PROOF. The proof relies on ProVerif for showing observational equivalences [Blanchet et al. 2005]. We automatically establish an equivalence by using a script derived from $\mathcal{S}^e$ of Appendix B.3, as explained below; the full definition of the script is available at `http://www.di.ens.fr/~blanchet/crypto/jfk.html`. Then we use standard pi calculus arguments to relate this equivalence to the one stated in the lemma. Our script is derived from the linear variant of $\mathcal{S}^e$ as follows:

(1) We replace R_{3a} with $\overline{d}\langle N_R, \mathsf{ID}_A\rangle \mid R_{3a}$, where d is an additional public channel. (This additional message reveals commitment to a Message 3 in a given receiver extended trace.) We also let $\mathcal{S}^{\circ\prime}$, $\mathcal{S}'[\_]$, and Q' be $\mathcal{S}^\circ$, $\mathcal{S}[\_]$, and Q with these additional messages.

(2) We replace $\nu exp.\nu cp$ with $\nu exp.\nu cp.\nu ids$ and add a replicated output $!\overline{ids}\langle K_-^A, \mathsf{ID}_A\rangle$ in parallel with $!\overline{cp}\langle \mathsf{ID}_A\rangle$.

(3) We obtain $\mathcal{S}_i^e$ for $i = 1, 2$ from the resulting script by adding within the scope of $\nu exp.\nu cp.\nu ids$ one of the additional processes T'_i defined as follows:

$$\begin{aligned} T'_i &= ids(K_-^A, \mathsf{ID}_A).ids(K_-^B, \mathsf{ID}_B).exp(d_{x_I}, x_I).exp(d_{x_R}, x_R).T_i \\ T_i &= init(\mathsf{ID}'_R, \mathsf{sa}_I, {=}\mathsf{ID}_A, {=}\mathsf{ID}_B, {=}x_I, {=}x_R). \\ &\quad \nu s_I, s_R, K_a, K_e.\nu N_I, N_R, t_R, K_v, e_I, e_R, h_I, h_R. \\ &\quad \overline{public}\langle N_I, N_R, t_R, K_v, e_I, e_R, h_I, h_R\rangle \mid \begin{cases} \varphi_I \mid \varphi_R \mid \kappa_I \mid \sigma_I \mid \sigma_R & \text{for } i = 1 \\ \varphi_4 & \text{for } i = 2 \end{cases} \end{aligned}$$

(4) We split the linear processes for the initiator and the responder by separating the signature computations from the rest of the processing: for the initiator, we split just after the computation of the signature s_I and before the computation of e_I, h_I and the sending of Message 3; for the responder, we move the signature computation before the MAC check, then split just before this check. We reassemble the resulting pieces as follows. For each principal $A \in \mathcal{C}$, we specify messages on new private channels $getI_A$ and $getR_A$ that carry a value for every variable used in the rest of the processing, we guard the rest of the processing with replicated inputs on these channels, and we place this (the rest of the processing plus the replicated inputs) in parallel with the remainders of $I^{A'}$ and $R^{A'}$ (which include signature computations). Let $\mathcal{S}_i^{e\prime}$ for $i = 1, 2$ be the resulting processes.

(5) The processes used in the ProVerif script for this proof are $\mathcal{S}_i^{e\prime\prime}$, where $\mathcal{S}_i^{e\prime} \equiv E[\mathcal{S}_i^{e\prime\prime}]$ for $i = 1, 2$ and some evaluation context E. (Intuitively, the context E collects the parts of $\mathcal{S}^e$ that do not affect the equivalence but complicate its automated proof; the splitting step (4) prepares this simplification.)

ProVerif automatically establishes $\mathcal{S}_1^{e\prime\prime} \approx \mathcal{S}_2^{e\prime\prime}$. Since equivalence is preserved by application of evaluation contexts, we have $\mathcal{S}_1^{e\prime} \approx \mathcal{S}_2^{e\prime}$ by applying E. After applying E, the auxiliary channels $getI_A$ and $getR_A$ are private, and used only to pass continuations to the rest of the initiator and responder processes, so we can undo Step (4) by using equivalences $\mathcal{S}_i^e \approx \mathcal{S}_i^{e\prime}$ for $i = 1, 2$. Hence we have $\mathcal{S}_1^e \approx \mathcal{S}_2^e$.

We apply the initialization trace $\mathcal{S}_i^e \xrightarrow{(\mathcal{C},X)} \mathcal{S}^e[T_i]$, as detailed in the proof of Lemma 4, followed by four communication steps leading from T_i' to T_i, for the parameters $\mathcal{C}$, X, ID_A, ID_B, x_I, and x_R used in the statement of the lemma. By bisimulation, and since this initialization trace determines the resulting processes $\mathcal{S}^e[T_i]$, we have $\mathcal{S}^e[T_1] \approx \mathcal{S}^e[T_2]$.

We now apply the context $\nu getexp, getprinc.[\_]$ then perform simplification steps:

—The channels $getprinc$, $getexp$, and cp are now restricted, and used only for output, so we can replace those outputs and their guarded processes with **0** up to equivalence.

—Exponents and exponentials are bound by parametric products $\prod_{x \in X}$ in $\mathcal{S}'[T_i]$, and are bound by inputs $exp(d_x, x)$ in $\mathcal{S}^e[T_i']$, in the presence of a replicated output $!\overline{exp}\langle d_x, x\rangle$ for each $x \in X$. Since exp does not appear anywhere else, the two variants are equivalent. We thus obtain the equivalence $\mathcal{S}'[T_1] \approx \mathcal{S}'[T_2]$.

We carry over the trace $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}[Q]$ in the hypothesis of the lemma, and obtain $\mathcal{S}'[T_i] \xrightarrow{\eta'} \mathcal{S}'[Q' \mid T_i]$ for $i = 1, 2$, where η' is η with additional outputs on d, and we finally use T_i:

$$\mathcal{S}'[Q' \mid T_i] \xrightarrow{\mathit{init}(\mathsf{ID}_R', \mathsf{sa}_I, x_I, x_R, \mathsf{ID}_A, \mathsf{ID}_B)} \xrightarrow{\nu N_I, N_R, t_R, K_v, e_I, e_R, h_I, h_R.\overline{public}\langle N_I, N_R, t_R, K_v, e_I, e_R, h_I, h_R\rangle} \begin{cases} \mathcal{S}'[Q' \mid \varphi] \\ \mathcal{S}'[Q' \mid \varphi_4] \end{cases}$$

By bisimulation, and since the actions on the trace determine the resulting processes, we still have $\mathcal{S}'[Q' \mid \varphi] \approx \mathcal{S}'[Q' \mid \varphi_4]$.

Finally, we apply the context $\nu d.\_$ and use the equivalence $\nu d.\mathcal{S}'[Q' \mid \varphi] \approx \mathcal{S}[Q \mid \varphi]$ (and similarly for φ_4). By transitivity, we conclude that $\mathcal{S}[Q \mid \varphi] \approx \varphi_4 \mid \mathcal{S}[Q]$. $\square$

C.2 Running Configurations: Transition Invariant

Partly relying on the static equivalences above to determine the outcome of tests, we can now relate extended traces and abstract configurations to arbitrary normal traces and their resulting configurations:

LEMMA 10. *Every normal trace $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}'$ is an interleaving of extended traces with final state $\mathcal{S}[Q]$ that matches $\mathcal{S}'$ up to structural equivalence and deterministic steps.*

PROOF. The proof is by induction on the length of the normal trace η. In the base case, the initial configuration coincides with the initial abstract configuration: $\mathcal{S}^\circ \equiv \mathcal{S}[\mathbf{0}]$.

For the inductive case, assume that we have a normal trace $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}' \xrightarrow{\alpha} \mathcal{S}''$. By inductive hypothesis, we can perform the case analysis on the final transition α on the abstract configuration $\mathcal{S}[Q]$ associated with η, instead of $\mathcal{S}'$. We enumerate all transitions enabled in $\mathcal{S}[Q]$ from its structure, given by Figure 3 and Definition 2:

—Replicated input on c in I_0, for some $A \in \mathcal{C}$ and $x_I \in X$. If the message matches the pattern $(\mathsf{ID}'_R, \mathsf{sa}_I)$ of I_0, then we create a new initiator extended trace in case (1) with action $\xrightarrow{\mathit{init}^A(\mathsf{ID}'_R,\mathsf{sa}_I)((x_I))}$. Otherwise, we use action $\xrightarrow{\mathit{init}^A(M)}$ to discard the message.

—Replicated input on c in R_1, for some $A \in \mathcal{C}$ and $x_I \in X$. If the message matches the input pattern of R_1, then we create a new responder extended trace, using case (2) with matching parameters. Otherwise we use action $\xrightarrow{c(M)}$ to discard the message.

—Output on c in I_1 for an initiator in stage (1). We move to the next stage, (2), by the same action. The cases for output on c in I_3, output on *connect*^{A} in I_{5a}, output on c in R_2, output on *accept*^{B} in R_{3c}, and output on c in R_4 are handled similarly.

—Input on c in I_2 for an initiator in stage (2). We move to stage (3) if the message matches the input pattern, and to stage ($\times$3) otherwise.

—Input on c in I_4 for an initiator in stage (4). We move to stage (5) if the two tests in I_4 and I_5 succeed, and to stage ($\times$5) otherwise.

—Replicated input on c in $R_3[\_]$ for a responder in stage $(3, n)$, (3a), ($\times$3a), (4), or (4′). In all these stages, we record the input message as $M_{3,n+1}$ and, if this message matches the pattern $3(=N_I, =N_R, x_I, =x_R, =t_R, e_I, h_I)$, we add a new process that inputs on l for $i = n + 1$. If we are at stage $(3, n)$, we move to stage $(3, n + 1)$; otherwise, we stay in the current stage. (Informally, the received message is discarded.)

—Replicated input on c in some process R'_3 for a responder in stage (3a), ($\times$3a), (4), or (4′). We extend R'_3 with a new deadlocked input on l if the message matches the third message pattern; the rest of the state is unchanged.

—Internal communication on channel l for a responder is stage $(3, n)$. We move to stage (3a) if its conditions are met, to stage ($\times$3a) otherwise. □

C.3 Proofs of the Theorems of Section 7 (Theorem 2, part of Theorem 4)

PROOF OF THEOREM 2. Assume that we have a normal trace $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$. By Lemma 2, we have $\mathcal{S} \approx \mathcal{S}^\circ$, hence $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}'$. By Lemma 10, we interpret this trace as an interleaving of extended transitions $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}''$ for some abstract state $\mathcal{S}'' = \mathcal{S}[Q]$ that matches $\mathcal{S}'$ up to structural equivalence and deterministic reduction steps.

From $\mathcal{S}[Q]$, the existence of the transitions made explicit in the statement of the theorem, and the characterization of the resulting configuration in terms of additional session states, are obtained as an interleaving of an initiator extended trace for A using x_I with a responder extended trace for B using x_R. From Definition 2 and Figure 3, we verify that each of the transitions is enabled in turn:

—When $\mathsf{ID}_A \in S_I^B$, we use the interleaving

$$\begin{aligned}\mathcal{S}[Q] \quad & \xrightarrow{\mathit{init}^A(\mathsf{ID}'_R,\mathsf{sa}_I)((x_I))} \xrightarrow{\nu N_I.\overline{c}\langle 1(N_I,x_I)\rangle} \xrightarrow{c(1(N_I,x_I))((B,x_R))} \\ & \xrightarrow{\nu N_R,t_R.\overline{c}\langle 2(N_I,N_R,x_R,\mathsf{g}_R,t_R)\rangle} \xrightarrow{c(2(=N_I,N_R,x_R,\mathsf{g}_R,t_R))} \\ & \xrightarrow{\nu e_I,h_I.\overline{c}\langle 3(N_I,N_R,x_I,x_R,t_R,e_I,h_I)\rangle} \xrightarrow{c(3(N_I,N_R,x_I,x_R,t_R,e_I,h_I))} \xrightarrow{((1))} \\ & \xrightarrow{\nu \mathsf{ID}_I,\mathsf{ID}'_R,\mathsf{sa}_I,K_v.\overline{\mathit{accept}}^B\langle \mathsf{ID}_I,\mathsf{ID}'_R,\mathsf{sa}_I,\mathsf{sa}_R,K_v\rangle} \\ & \xrightarrow{\nu e_R,h_R.\overline{c}\langle 4(e_R,h_R)\rangle} \xrightarrow{c(4(e_R,h_R))} \\ & \xrightarrow{\nu \mathsf{ID}_R,\mathsf{sa}_R,K_v.\overline{\mathit{connect}}^A\langle \mathsf{ID}_R,\mathsf{ID}'_R,\mathsf{sa}_I,\mathsf{sa}_R,K_v\rangle} \quad \mathcal{S}[Q \mid {I'_5}^A \mid {R'_4}^B \mid R'_3]\end{aligned}$$

By definition, R'_3 is inert in this context, and can be discarded up to equivalence. Finally, we apply the equivalence of Lemma 9 to replace ${I'_5}^A \mid {R'_4}^B$ with φ_4.

—When $\mathsf{ID}_A \notin S_I^B$, we use the interleaving

$$\begin{aligned}\mathcal{S}[Q] \quad & \xrightarrow{\mathit{init}^A(\mathsf{ID}'_R,\mathsf{sa}_I)((x_I))} \xrightarrow{\nu N_I.\overline{c}\langle 1(N_I,x_I)\rangle} \xrightarrow{c(1(N_I,x_I))((B,x_R))} \\ & \xrightarrow{\nu N_R,t_R.\overline{c}\langle 2(N_I,N_R,x_R,\mathsf{g}_R,t_R)\rangle} \xrightarrow{c(2(=N_I,N_R,x_R,\mathsf{g}_R,t_R))} \\ & \xrightarrow{\nu e_I,h_I.\overline{c}\langle 3(N_I,N_R,x_I,x_R,t_R,e_I,h_I)\rangle} \xrightarrow{c(3(N_I,N_R,x_I,x_R,t_R,e_I,h_I))} \xrightarrow{((1))} \\ & \mathcal{S}[Q \mid (\varphi_I \mid \nu K_a, K_e, K_v, s_I.(\kappa_I \mid \sigma_I \mid I_4^A)) \mid (\varphi_R \mid R_3^B)]\end{aligned}$$

The simplification of the resulting process is obtained as a corollary of Lemma 9: the equivalence for complete, accepted sessions implies the corresponding equivalence for complete, rejected sessions, by applying a context that restricts variables not exported in the latter case. Finally, the tests in subprocesses I_4^A and R_3^B always fail, so these processes can be discarded up to bisimilarity (since any abstract configuration can receive and discard messages on c anyway). □

PROOF OF SECOND POINT OF THEOREM 4. By Lemma 10, the normal trace in the statement of the theorem can be decomposed into an interleaving of extended traces. Since there is a *connect*^{A} action for some $A \in \mathcal{C}$, the trace comprises (at least) one initiator extended trace for A in case (5′) with that action. Using the first point of Theorem 4 (verified by ProVerif), there exists an *accept*^{B} action for some $B \in \mathcal{C}$ with matching parameters, hence η comprises (at least) one responder extended trace for B in case (4) or (4′) with that action. Hence, we have

$$\mathcal{S}^\circ \xrightarrow{\eta} \xrightarrow{\overline{\mathit{connect}}^A\langle \mathsf{ID}_B,\mathsf{ID}'_R,\mathsf{sa}_I,\mathsf{sa}_R,K_v\rangle} \mathcal{S}[Q \mid I'_5 \mid R'_4 \mid R'_3] \tag{7}$$

where I'_5 and $R'_4 \mid R'_3$ result from those two extended traces and Q corresponds to all other extended traces. We check that the two extended traces share the same parameters, so as to match the series of actions listed in Theorem 4:

—Matching parameters between *accept*^{B} and *connect*^{A} actions directly include the terms $\mathsf{ID}_R = \mathsf{ID}_B$, $\mathsf{ID}_A = \mathsf{ID}_I$, ID'_R, sa_I, sa_R, and K_v.

—The key computations κ_I and κ_R of Definition 2 yield equal terms if and only if they use matching parameters N_I, N_R, x_I, and x_R. So matching keys K_v implies matching terms N_I, N_R, x_I, x_R, K_a, and K_e.

—By inspection of abstract states in Definition 2, signatures under K_-^A use distinct names as first signed parameter, so there is only one such verifiable signature for N_I. Similarly, there is only one verifiable signature under K_-^B with N_R as second signed parameter. Thus, the sent and received signatures coincide, and moreover the responder extended trace must have exported this signature and therefore be in case (4′).

—The remaining terms e_I, h_I, e_R, and h_R depend only on matching terms listed above.

Since I'_5 and $R'_4 \mid R'_3$ share the same parameters and R'_3 is inert in this context, we have $\mathcal{S}[Q \mid I'_5 \mid R'_4 \mid R'_3] \approx \mathcal{S}[Q \mid \varphi]$. By Lemma 9, we have $\mathcal{S}[Q \mid \varphi] \approx \varphi_4 \mid \mathcal{S}[Q]$. Finally, let η' be η minus the actions for the two extended traces. Relying on our analysis of the transitions (7) as an interleaving of extended traces, we obtain $\varphi_4 \mid \mathcal{S}^\circ \xrightarrow{\eta'} \varphi_4 \mid \mathcal{S}[Q]$. □

C.4 Proof of the Theorem of Section 8 (Theorem 5)

In order to prove our plausible deniability results, we explain how to simplify the frame associated with an abstract configuration, so as to replace most of the terms defined in past sessions by terms that do not use any name restricted in $\mathcal{S}[\_]$ (signing keys and secret exponents).

The next lemma describes how an active attacker can get access to keys and signatures by running ordinary sessions with compliant principals.

LEMMA 11 (TRANSPARENT SESSIONS). *We say that an extended trace is* transparent *when the peer exponential provided by the environment (x_R in Message 2 for the initiator, x_I in Message 3 for the responder) equals $g\,\hat{}\,D$ for some term D.*

(*1*) *Let $\mathcal{S}[Q \mid I'_5]$ be an abstract configuration where I'_5 is the state of a transparent extended trace in case (5′), which defines $\sigma_{s_I} = \{s_I = \mathsf{S}\{K^A_-\}(N_I, N_R, x_I, x_R, \mathsf{g}_R)\}$. There are evaluation contexts E and F that do not restrict the variables of $\mathcal{S}[Q]$ such that*

$$\mathcal{S}[Q \mid I'_5] \equiv E[\mathcal{S}[Q \mid \varphi_I \mid \sigma_{s_I}]] \quad \textit{and} \quad \mathcal{S}[Q \mid \varphi_I \mid \sigma_{s_I}] \equiv F[\mathcal{S}[Q \mid I'_5]]$$

(*2*) *Let $\mathcal{S}[Q \mid R'_4]$ be an abstract configuration where R'_4 is the state of a transparent extended trace in case (4′), which defines $\sigma_{s_R} = \{s_R = \mathsf{S}\{K^A_-\}(N_I, N_R, x_I, x_R)\}$. There are evaluation contexts E and F that do not restrict the variables of $\mathcal{S}[Q]$ such that*

$$\mathcal{S}[Q \mid R'_4] \equiv E[\mathcal{S}[Q \mid \varphi_R \mid \sigma_{s_R}]] \quad \textit{and} \quad \mathcal{S}[Q \mid \varphi_R \mid \sigma_{s_R}] \equiv F[\mathcal{S}[Q \mid R'_4]]]$$

PROOF. We give a proof of the first part of the lemma; the other part is established similarly. Relying on the Diffie-Hellman equation and structural equivalence, we replace each occurrence of the keys K_u for $u = a, e, v$ defined by κ_I with the corresponding term $\mathsf{H}\{x_I\,\hat{}\,D\}(N_I, N_R, \mathrm{u})$. Hence, for some κ'_I, τ'_I, and σ'_{e_I,h_I} variants of κ_I, τ_I, and σ_I for e_I, h_I that do not use names restricted by $\mathcal{S}[\_]$, we have

$$\begin{aligned}\mathcal{S}[Q \mid I'_5] \;&=\; \mathcal{S}[Q \mid \varphi_I \mid \nu s_I, s_R, K_a, K_e.(\kappa_I \mid \sigma_I \mid \tau_I)]\\ &\equiv\; \nu s_I, s_R, K_a, K_e.\big(\kappa'_I \mid \tau'_I \mid \sigma'_{e_I,h_I} \mid \mathcal{S}[Q \mid \varphi_I \mid \sigma_{s_I}]\big)\end{aligned}$$

Conversely, we use $F[\_] = \nu e_I, K_e.(\kappa_{K_e} \mid \tau_{R,s_I} \mid \nu K_v, \mathsf{ID}_R, \mathsf{sa}_R, s_R, h_I.\_)$ where κ_{K_e} defines K_e as in κ'_I and τ_{R,s_I} defines s_I as in τ_R of Figure 3. □

Next we deal with non-transparent extended traces. Sessions with identical parameters N_I, N_R, x_I, and x_R can be simplified by using Lemma 9. The next lemma deals with the key computation for any other sessions.

LEMMA 12. *For a given abstract configuration, let η_I and η_R range over non-transparent extended traces with different tuples of terms (N_I, N_R, x_I, x_R) for any two extended traces. Let $\kappa' = \prod_{u=a,e,v} \nu N.\{K_u = N\}$. We have:*

$$D_X\left[\textstyle\prod_{\eta_I}(\varphi_I \mid \kappa_I) \mid \prod_{\eta_R}(\varphi_R \mid \kappa_R)\right] \;\approx_s\; D_X\left[\textstyle\prod_{\eta_I}(\varphi_I \mid \kappa') \mid \prod_{\eta_R}(\varphi_R \mid \kappa')\right]$$

PROOF. The active substitutions κ_I and κ_R in the session states define keys $K_{I,u} = \mathsf{H}\{x_R\,\hat{}\,d_{x_I}\}(N_I, N_R, \mathrm{u})$ and $K_{R,u} = \mathsf{H}\{x_I\,\hat{}\,d_{x_R}\}(N_I, N_R, \mathrm{u})$. Since $\mathsf{H}\{\_\}(\_)$ has no equation and at least some of the parameters used as second argument are different in any two of these definitions, these keys are pairwise distinct. Since the extended traces are not transparent, the variable x_R is bound either to $x \in X$ or to a term that is not an exponential. In both cases, the computed Diffie-Hellman term $x_R\hat{}d_{x_I}$ is different from any term available to the environment, so each key is also different from any other term available to the environment. $\square$

We can now rewrite the frame associated with any abstract configuration to a frame that exports signatures on transparent extended traces, plus distinct fresh names:

LEMMA 13. *For any normal trace $\mathcal{S}^\circ \xrightarrow{\eta} \mathcal{S}'$, we have $\mathcal{S}' \approx_s \Phi \mid \mathcal{S}[Q']$ where Φ is an active substitution that exports fresh, distinct names and Q' is a parallel composition of the states associated with transparent extended traces in the abstract configuration $\mathcal{S}'$.*

PROOF. Let $\mathcal{S}[Q] \approx \mathcal{S}'$ be the abstract configuration provided by Lemma 10. We apply Lemma 9 to every pair of extended traces with matching parameters N_I, N_R, x_I, x_R, thus rewriting their state into a frame that defines fresh, distinct names, then collect all session states that define only fresh, distinct names into a single active substitution Φ.

Let Q' and Q'' be the parallel compositions of the states for all remaining transparent and non-transparent extended traces, respectively. By applying Lemma 11 to every transparent extended trace, we can rewrite Q' into Q'_E where none of the secret exponents $(d_x)_{x\in X}$ occur. Hence, we have

$$\mathcal{S}[Q] \approx_s \Phi \mid \mathcal{S}[Q' \mid Q''] \equiv \Phi \mid PK^{\mathcal{C}}\,[Q'_E \mid D_X[Q'']]$$

We apply Lemma 12 (in some evaluation context) to simplify $D_X[Q'']$ so as to replace all key computations κ_R and κ_I within Q'' with instances of κ'. Then, using simple static equivalences, we further simplify the state for all non-transparent extended traces by replacing any exported terms e_I, e_R, h_I, h_R with exported fresh names (since each of these terms is keyed with the only occurrence of a restricted name). Let Φ' collect all the resulting fresh, distinct name definitions. We thus have

$$\Phi \mid PK^{\mathcal{C}}\,[Q'_E \mid D_X[Q'']] \approx_s \Phi \mid \Phi' \mid PK^{\mathcal{C}}\,[Q'_E \mid D_X[\mathbf{0}]] \equiv \Phi \mid \Phi' \mid \mathcal{S}[Q'] \quad \square$$

PROOF OF THEOREM 5. For each transformation described in the theorem, we verify that the normal trace $\mathcal{S}_a \xrightarrow{\eta_a} \mathcal{S}'_a$ is enabled, describe the effect of the transformation on the frame obtained by Lemma 13, and exhibit an evaluation context C that does not restrict the variables of $\mathcal{S}_a$ such that $C[\mathcal{S}'_a] \approx_s \mathcal{S}'$.

(1) The existence of the trace is given by the second part of Theorem 4. After applying Lemma 13, sessions between compliant principals only export fresh, distinct names. Let φ_4 define these extra names; we use the context $C[\_] = \varphi_4 \mid [\_]$.
(2) Since A performs no control action, no session associated with A produces any signature. We erase from η any session associated with A, let φ define instead fresh, distinct names, and let C be the context $PK^A\,[\mathbf{0}] \mid \varphi \mid [\_]$.
(3) Those parameters do not affect the existence of the trace and, if they occur in non-transparent extended traces, they are erased by Lemma 13. Otherwise, we detail the case (5′): relying on Lemma 11, we let $C[\_] = E[F_a[\_]]$ where E uses the parameters of the original extended trace and F_a is F with the rewritten parameters.
(4) We erase the *connect*^{A} from the trace and obtain a session in case (5) instead of (5′). Relying on the first part of Lemma 11, we let $C[\_] = E[F'[\_]]$, where F' is F without the restriction on K_v, ID_R, and sa_R.

(5) Those parameters do not affect the existence of the trace. In particular, for transparent extended traces for the responder, the hypothesis on ID_I guarantees that the environment can provide a valid Message 3 that includes a signature associated with the rewritten verification key. We detail the case (4′): relying on the second part of Lemma 11, we let $C[\_] = E[F_a[\_]]$ where E uses the parameters of the original extended trace and F_a is F with the rewritten parameters.

(6) This change affects neither the existence of the trace nor the frame obtained after applying Lemma 13. □

Received September 2005; revised October 2006; accepted December 2006.