Just Fast Keying in the Pi Calculus

Martín Abadi[1], Bruno Blanchet[2], and Cédric Fournet[3]

[1] University of California, Santa Cruz
[2] CNRS, Département d'Informatique, École Normale Supérieure, Paris
and Max-Planck-Institut für Informatik, Saarbrücken
[3] Microsoft Research

Abstract. 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 resistance to denial-of-service attacks. In the course of this analysis we found some ambiguities and minor problems, 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.

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 [12], 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 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 [5, 6] (for "just fast keying") and the IKEv2 protocol [13], currently under development.

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 one of the two variants of JFK known as JFKr (the one closer to IKEv2). 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 several properties of the protocol, 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 [7]. 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.

Contents The next section is a review and informal discussion of JFK. Section 3 presents a model of JFKr in the applied pi calculus. Section 4 treats DOS resistance. Section 5 concerns core security properties (secrecy and authenticity). It also briefly addresses identity protection. Section 6 mentions some related work and concludes.

Because of space constraints, this version of the paper omits much material that appears in an extended version [3]: a discussion of the ambiguities and minor problems that we found, a description and partial analysis of the protocol variant JFKi, a review of the applied pi calculus, further material on identity protection and on DOS resistance, a study of plausible deniability, details on our use of ProVerif, and proofs.

2 The JFK Protocol

The JFK protocol has been discussed in a series of five Internet Drafts [5], starting in 2001, and it is also described in a conference paper [6]. While our work is based on all those documents, we tend to privilege the contents of the conference paper, since it should have some permanence. We refer to that paper for additional material on the protocol and its motivation. As indicated above, we focus on a variant called JFKr.

JFKr involves two principals that play the roles of an initiator and a responder. 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 [19]. Therefore, the communications between the initiator and the responder are cryptographically protected. Informally, JFKr consists of the following four messages:

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

where:

$$
\begin{array}{ll}
x_I = \mathsf{g}\hat{}d_I & x_R = \mathsf{g}\hat{}d_R \\
 & t_R = \mathsf{H}\{K_R\}(x_R, N_R, N_I, \mathsf{IP}_I) \\
\multicolumn{2}{l}{K_u = \mathsf{H}\{x_R\hat{}d_I\}(N_I, N_R, \mathsf{u}) \quad \text{for } u = a, e, v} \\
e_I = \mathsf{E}\{K_e\}(\mathsf{ID}_I, \mathsf{ID}'_R, \mathsf{sa}_I, s_I) & e_R = \mathsf{E}\{K_e\}(\mathsf{ID}_R, \mathsf{sa}_R, s_R) \\
h_I = \mathsf{H}\{K_a\}(\mathsf{i}, e_I) & h_R = \mathsf{H}\{K_a\}(\mathsf{r}, e_R) \\
s_I = \mathsf{S}\{K_I^-\}(N_I, N_R, x_I, x_R, \mathsf{g}_R) & s_R = \mathsf{S}\{K_R^-\}(x_R, N_R, x_I, N_I)
\end{array}
$$

Figure 1 summarizes the notations of this exchange.

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, and need not perform any expensive cryptography or allocate resources until such a successful round-trip with the initiator. 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.

$z = I, R$	one of the two roles in the protocol: initiator or responder.
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 [6]).
g	Diffie-Hellman group (possibly obtained from a previously received g_R).
g_R	responder's choice of group g and cryptographic algorithms ($GRPINFO_R$ in [6]).
t_R	authenticator cookie used by the responder against DOS.
K_R	responder's secret hash key for authenticators t_R (HK_R in [6]).
$u = a, e, v$	one of the three usage for keys: authentication, encryption, and main session secret.
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 setting IP security associations (sa and sa' in [6]).
A, B	principals, taking part in the protocol (in either or both roles).

Fig. 1. Main notations

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 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.

The protocol specification, although clear, focuses on the messages exchanged in a single successful protocol run. 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 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 and study them formally below.

3 A Model of JFK in the Applied Pi Calculus

In this section, we express JFKr in the applied pi calculus. 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. (We refer to prior work [4] for its

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

$(\mathsf{g}\,\hat{}\,y)\,\hat{}\,z = (\mathsf{g}\,\hat{}\,z)\,\hat{}\,y$	Diffie-Hellman
$\mathsf{V}\{\mathsf{Pk}(k), \mathsf{S}\{k\}(x)\}(x) = \mathsf{true}$	public-key signature verification
$\mathsf{D}\{k\}(\mathsf{E}\{k\}(x)) = x$	shared-key decryption
$\mathsf{F}_i^{\mathsf{n}}(\mathsf{n}(x_1, \ldots, x_i, \ldots)) = x_i$	projections for tuples ($\mathsf{n} = \mathsf{cons}, 1, 2, 3, 4$)
$(\emptyset.x).x = \emptyset.x$	idempotence of set extension
$(x.y).z = (x.z).y$	commutativity of set extension
$\mathsf{RecoverKey}(\mathsf{S}\{k\}(x)) = \mathsf{Pk}(k)$	public key recovery from a signature (attacker)
$\mathsf{RecoverText}(\mathsf{S}\{k\}(x)) = x$	signed text recovery from a signature (attacker)

Fig. 2. Grammar and equational theory for JFK

syntax and semantics.) So we first select function symbols and an equational theory for modelling the messages of JFKr, then we discuss our representations for the IP network, attackers, and principals, and assemble processes that represent configurations of principals. We also outline how we program these processes in ProVerif.

An Equational Theory We use the grammar for terms and the equations of Figure 2. These deal both with cryptographic operations and with auxiliary functions for constructing tags, pairs, formatted messages, and sets. (We have functions for constructing sets, but not a set membership relation; instead, we let $U \in V$ abbreviate $V.U = V$.)

The equations embody our (fairly standard) hypotheses on the primitives introduced in Section 2. For instance, the keyed hash function $\mathsf{H}\{\_\}(\_)$ does not appear in any equation, and in particular has no inverse; thus it represents a perfect one-way function. More interestingly, exponentiation $\_\,\hat{}\,\_$ has no inverse, but an equation accounts for the commutativity property used for establishing a shared secret. 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: $\mathsf{RecoverKey}(\_)$ and $\mathsf{RecoverText}(\_)$ can be used to extract information from signatures. We could further refine our theory by reflecting known weaknesses of the underlying cryptographic algorithms or their interactions.

Syntax and Informal Semantics for Processes We recall the main notations for processes in the applied pi calculus : **0** does nothing (and we typically omit it); $P \mid Q$ is the parallel composition of P and Q; $!P$ behaves as an infinite number of copies of P run-

ning in parallel; $\nu n.P$ makes a new name n then behaves as P; *if* $U = V$ *then* P *else* Q is standard, with $U = V$ depending on the equational theory. The input process $c(x).P$ is ready to input a message on channel c, then to run P with the actual message replaced for the formal parameter x, while the output process $\overline{c}\langle V\rangle.P$ is ready to output message V on channel c, then to run P. The process *let* $\{x_1 = V_1\} \mid \ldots \mid \{x_n = V_n\}$ *in* P is P with local variables $x_1, \ldots, x_n$ bound to $V_1, \ldots, V_n$, respectively. The active substitution $\{x_1 = V_1\} \mid \ldots \mid \{x_n = V_n\}$ similarly defines $x_1, \ldots, x_n$, but does not restrict their scope; hence, the environment can use $x_1, \ldots, x_n$ as aliases for $V_1, \ldots, V_n$ in its computations. A context $C[\_]$ is a process with a hole, and $C[P]$ is the result of filling $C[\_]$'s hole with P; when $\_$ is not under a guard, $C[\_]$ is an evaluation context.

Labelled transitions $P \xrightarrow{a(V)} Q$ and $P \xrightarrow{\nu\widetilde{u}.\overline{a}\langle V\rangle} Q$ represent interactions with the environment—inputs and outputs, respectively. In both, a is a communication channel and V a message. Transitions $P \rightarrow Q$ represent internal computation steps.

Syntactic Sugar We write *if* M *then* P instead of *if* $M = \mathsf{true}$ *then* P. 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 $c(z).let\ \{x_I = \mathsf{F}_2^1(z)\}\ in\ if\ z = 1(N_I, x_I)\ then\ P$ 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. We also define syntax for filtering duplicate messages: $!a(X)\backslash V.C[if\ T\ fresh\ then\ P]$ stands for

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

where $C[\_]$ is a context, X is a pattern, f is a fresh channel name, and s is a fresh variable. We use the local channel f 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.

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 model 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 (formally, with an output label $\nu\widetilde{u}.\overline{c}\langle V\rangle$) followed by a re-emission of the same message (with an input label $c(V)$). We write $P \xrightarrow{\nu\widetilde{u}.c[V]} P'$ as a shorthand for the two transitions $P \xrightarrow{\nu\widetilde{u}.\overline{c}\langle V\rangle} \xrightarrow{c(V)} P'$.

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 $PK^A[\_]$ of Figure 3 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. Whenever we put a process R in this context, our intent is that R never communicates K_A^- to the environment.

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.

Our representation of a compliant principal A has two parts: an implementation of JFKr, written $\mathcal{S}$, and 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. The 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.

Thus, the interface between each principal A and JFKr consists of three communication channels init_A, accept_A, $\mathit{connect}_A$ plus a set of identities S_I^A. These channels can be restricted (with ν) in order to hide the interface from the environment. For instance, a principal A that does not play the role of an initiator can be modelled easily by restricting communication on init^A.

The Protocol Figure 3 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 detail the tests performed on incoming messages, interleaved with the computations on outgoing messages. 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.

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

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

$$
\begin{array}{rll}
R_1^A = & !c(\mathsf{1}(N_I, x_I)). & \textbf{Responder for each Message 1} \\
& \nu N_R. & \textit{create a fresh nonce} \\
& \mathit{let}\ \{t_R = \mathsf{H}\{K_R\}(x_R, N_R, N_I)\}\ \mathit{in} & \textit{compute anti-DOS token} \\
& \overline{c}\langle \mathsf{2}(N_I, N_R, x_R, \mathsf{g}_R, t_R)\rangle & \textit{send Message 2}
\end{array}
$$

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

$$
\begin{array}{rll}
\mathcal{S} = & D_X\left[\prod_{A\in\mathcal{C}} PK^A\left[I^A|R^A\right]\right] & \textbf{Compliant principal configuration} \\
I^A = & \prod_{x_I\in X} I_0^A & A\ \textit{as initiator} \\
R^A = & \nu K_R.\prod_{x_R\in X}(R_1^A \mid R_3^A) & A\ \textit{as responder}
\end{array}
$$

$$
PK^A[\_] = \nu K_A^-.(\{\mathsf{ID}_A = \mathsf{Pk}(K_A^-)\} \mid [\_]) \qquad A\text{'s signing and verification keys}
$$

$$
\begin{array}{rll}
D_x[\_] = & \nu d_x.(\{x = \mathsf{g}\hat{}d_x\} \mid [\_]) & \textit{DH secret } d \textit{ and exchange value } x \\
D_X[\_] = & D_{x_1}[\ldots D_{x_n}[\_]] \text{ where } X = \{x_1, \ldots, x_n\} & \textit{shared exponentials} \\
\kappa_I = & \prod_{u=a,e,v}\{K_u = \mathsf{H}\{x_R\hat{}d_{x_I}\}(N_I, N_R, \mathsf{u})\} & \textit{key computations for } I \\
\kappa_R = & \prod_{u=a,e,v}\{K_u = \mathsf{H}\{x_I\hat{}d_{x_R}\}(N_I, N_R, \mathsf{u})\} & \textit{key computations for } R
\end{array}
$$

Fig. 3. JFKr in the applied pi calculus

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. 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.

Script for Proof Automation We rely at least partially on ProVerif in most proofs. For that purpose, we code JFK configurations ($\mathcal{S}$ in Figure 3) in the input syntax of ProVerif (which is an ASCII syntax for the applied pi calculus), specify the properties to prove, and simply run ProVerif. The correctness of these proofs relies on the theory developed in prior work ([7, 1] for secrecy, [8] for correspondence assertions, [2] for some extensions). Additional details on ProVerif and our script appear in the full paper [3]. The script differs superficially from $\mathcal{S}$ in that it 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 $connect^A$.

4 Resistance to Denial-of-Service Attacks

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. Our formal analysis relies on an understanding of the costs incurred at these stages: 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 depends 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 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 variables $\widetilde{x}_2$, let K be a fresh name, let $P = !c(x_3)$, and*

$$R_2^\circ = \nu N.\nu t.\overline{c}\langle M_2\rangle.?c(\mathsf{3}(=t, =N, =\widetilde{x}_2, \widetilde{x}_3)).R_4$$
$$R_2 = \nu N.\mathit{let}\ \{t = \mathsf{H}\{K\}(N, \widetilde{x}_2)\}\ \mathit{in}\ \overline{c}\langle M_2\rangle$$
$$R_3 = !c(\mathsf{3}(t, N, \widetilde{x}_2, \widetilde{x}_3))\setminus \emptyset.\mathit{if}\ t = \mathsf{H}\{K\}(N, \widetilde{x}_2)\ \mathit{then\ if}\ t\ \mathit{fresh\ then}\ R_4$$

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

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 $?c(X).R$ for $\nu l.(\overline{l}\langle\rangle \mid !c(X).l().R)$. (In our automated proofs, we need to use an equivalent but 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.)

Informally, $N, \widetilde{x}_2$ represents the state of the protocol at the end of R_2 that is used later in R_4, M_2 represents a message carrying (at least) N and t, and $\widetilde{x}_3$ represents new data received in Message 3. The presence of the same state $N, \widetilde{x}_2$ 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 a message x_3 and discarding it after a failed test. 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.

The next lemma applies this protocol transformation to JFKr. It relates our main model $\mathcal{S}$ (see Figure 3), 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 $R_1^A \mid R_3^A$ in each R^A by*

$$\begin{aligned} R_1^{\circ A} = {} & !c(1(N_I, x_I)).\nu N_R, t_R.\overline{c}\langle 2(N_I, N_R, x_R, \mathsf{g}_R, t_R)\rangle. \\ & ?c(3(=N_I, =N_R, x_I, =x_R, =t_R, e_I, h_I)). \\ & \textit{let } \kappa_R \textit{ in } \ldots \qquad \textit{(as in } R_3^A\textit{)} \end{aligned}$$

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. 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 [19]) may send Messages 1 with fake IP addresses, but then may not be able to eavesdrop on the corresponding Messages 2.

Theorem 1 (Protection from DOS). *Let $A \in \mathcal{C}$, and let $\mathcal{S}_\$$ be $\mathcal{S}$ with an additional output $\overline{\$}\langle N_I, N_R\rangle$ before the Diffie-Hellman computation κ_R in R_3^A. For any trace $\mathcal{S}_\$ \xrightarrow{\eta} \mathcal{S}'$, for each output $\overline{\$}\langle N_I, N_R\rangle$, there are distinct, successive actions $c(1(N_I, \_))$, $\overline{c}\langle 2(N_I, N_R, \_, \_, \_)\rangle$, and $c(3(N_I, N_R, \_, \_, \_, \_, \_))$.*

The additional output on $\$$ serves as a marker for the start of expensive processing (public-key operations and session-state allocation). The theorem formulates "round-trip authentication" as an injective correspondence between actions. The correspondence depends on the authenticator; we prove it by applying a variant of Lemma 2 to obtain an equivalent, linear protocol, and invoking ProVerif on that protocol. In incorrect interpretations of JFKr, Theorem 1 is false, and many Messages 3 may be processed for the same authenticator [3].

5 Core Security: Secrecy and Authenticity

Next, we consider session-key secrecy and mutual authentication. 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, 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

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

and $\xrightarrow{[4]}$ 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. (A frame is basically an active substitution with 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). *Assume $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$. For any 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*

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

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

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

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

This theorem 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. It 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 plain names ($\nu N.\{x = N\}$), our representation of independent, fresh values in the pi calculus. 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), with an active attacker.

Theorem 3 (Authenticity for Control Actions). *Assume $\mathcal{S} \xrightarrow{\eta} \mathcal{S}'$. The actions in η have the following properties:*

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

The proof of these properties relies on ProVerif. 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 trace includes a control message $\overline{\mathit{connect}}^A\langle \mathsf{ID}_B, \dots \rangle$ for some $A, B \in \mathcal{C}$, the trace essentially contains a normal successful run of the protocol with an eavesdropper, as described in Theorem 2. The hypotheses exclude internal communication on c; this assumption means that the attacker sees all messages on c, and is convenient but not essential.

Theorem 4 (Authenticity for Complete Sessions). *Let $A, B \in \mathcal{C}$ and assume*

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

without internal communication steps on c, and with fresh, distinct exported variables.

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 use ProVerif to show the first point of this theorem, via Lemma 2.

Theorem 3 is simpler and more abstract than Theorem 4, as it deals 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 [3]. The absence of authentication of t_R by the initiator seems harmless, inasmuch as t_R is used only by R.

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 [4], 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 also shows 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 presupposes, then verifies, that the environment never obtains signing keys K_A^- for $A \in \mathcal{C}$ or Diffie-Hellman secrets d_x for $x \in X$.)

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 [11]). 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. We can prove additional identity-protection properties, in some cases with ProVerif; on the other hand, we have also found some limitations in identity protection [3].

6 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 paper that presents JFK argues informally about some of its core properties, and calls for a formal analysis. Recent work by Datta et al. explores how the STS protocol, two JFK variants, and the core of IKE can be derived by successive refinements [9, 10]. In particular, it isolates the usage of authenticators and discusses the properties of JFKr (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) [17, 7, 8], and a framework for the study of DOS [18]. More broadly, the literature contains several formal techniques for protocol analysis and many examples (e.g., [14, 16, 20, 21, 15]). 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, with a reasonable effort; 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.

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 is relatively recent and, at the time of this writing, it continues to evolve, with influence from JFK and other sources. At present, JFK and its specification seem inherently more self-contained and interesting than IKEv2. On the other hand, 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 was partly done at Microsoft Research, Silicon Valley, and it was also partly supported by the National Science Foundation under Grants CCR-0204162 and CCR-0208800.

References

1. M. Abadi and B. Blanchet. Analyzing security protocols with secrecy types and logic programs. In *29th ACM SIGPLAN - SIGACT Symposium on Principles of Programming Lan-*

guages (POPL'02), pages 33–44, Jan. 2002.
2. M. Abadi and B. Blanchet. Computer-assisted verification of a protocol for certified email. In *Static Analysis, 10th International Symposium (SAS'03)*, volume 2694 of *LNCS*, pages 316–335. Springer-Verlag, June 2003.
3. M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus. Manuscript, available from `http://www.di.ens.fr/~blanchet/crypto/jfk.html`, Dec. 2003.
4. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, Jan. 2001.
5. W. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ionnidis, A. Keromytis, and O. Reingold. Just fast keying (JFK). IETF Internet Draft `draft-ietf-ipsec-jfk-04.txt`, July 2002.
6. W. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ionnidis, A. Keromytis, and O. Reingold. Efficient, DoS-resistant, secure key exchange for internet protocols. In *9th ACM Conference on Computer and Communications Security (CCS'02)*, pages 48–58, Nov. 2002.
7. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, June 2001.
8. B. Blanchet. From secrecy to authenticity in security protocols. In *Static Analysis, 9th International Symposium (SAS'02)*, volume 2477 of *LNCS*, pages 342–359. Springer-Verlag, Sept. 2002.
9. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *16th IEEE Computer Security Foundations Workshop (CSFW-16)*, pages 109–125, July 2003.
10. A. Datta, J. C. Mitchell, and D. Pavlovic. Derivation of the JFK protocol. `http://www.stanford.edu/~danupam/composition.ps`, 2002.
11. C. Fournet and M. Abadi. Hiding names: Private authentication in the applied pi calculus. In *Software Security – Theories and Systems. Mext-NSF-JSPS International Symposium (ISSS'02)*, volume 2609 of *LNCS*, pages 317–338. Springer-Verlag, Jan. 2003.
12. D. Harkins and D. Carrel. RFC 2409: The Internet Key Exchange (IKE). `http://www.ietf.org/rfc/rfc2409.txt`, Nov. 1998.
13. Internet Key Exchange (IKEv2) Protocol. IETF Internet Draft at `http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-11.txt`, Oct. 2003.
14. R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, Spring 1994.
15. P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Fifth ACM Conference on Computer and Communications Security (CCS'98)*, pages 112–121, 1998.
16. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, 1996.
17. C. Meadows. Analysis of the Internet Key Exchange protocol using the NRL protocol analyzer. In *IEEE Symposium on Security and Privacy*, pages 216–231, May 1999.
18. C. Meadows. A cost-based framework for analysis of denial of service networks. *Journal of Computer Security*, 9(1/2):143–164, 2001.
19. R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, Dec. 1978.
20. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1–2):85–128, 1998.
21. F. J. Thayer Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *IEEE Symposium on Security and Privacy*, pages 160–171, May 1998.