Just Fast Keying: Key Agreement in a Hostile Internet

WILLIAM AIELLO, STEVEN M. BELLOVIN, MATT BLAZE
AT&T Labs Research
RAN CANETTI
IBM T. J. Watson Research Center
JOHN IOANNIDIS
AT&T Labs Research
ANGELOS D. KEROMYTIS
Columbia University
and
OMER REINGOLD
AT&T Labs Research

We describe Just Fast Keying (JFK), a new key-exchange protocol, primarily designed for use in the IP security architecture. It is simple, efficient, and secure; we sketch a proof of the latter property. JFK also has a number of novel engineering parameters that permit a variety of tradeoffs, most notably the ability to balance the need for perfect forward secrecy against susceptibility to denial-of-service attacks.

Categories and Subject Descriptors: C.2.0 [**Security and Protection**]: Key Agreement Protocols

General Terms: Security, Reliability, Standardization

Additional Key Words and Phrases: Cryptography, denial-of-service attacks

1. INTRODUCTION

Many public-key-based key-setup and key-agreement protocols already exist and have been implemented for a variety of applications and environments. Several have been proposed for the IPsec protocol suite, and one, internet key exchange (IKE) [Harkins and Carrel 1998], is the current standard.

This work was partially supported by DARPA under contract F39502-99-1-0512-MOD P0001. A previous version of this paper appeared in Aiello et al.[2003].
Author's addresses: William Aiello, Steven M. Bellovin, Matt Blaze, John Ioannidis, Omer Reingold, AT&T Labs Research, email: {aiello,smb,mab,ji,omer}@research.att.com; Ran Canetti, IBM T.J. Watson Research Center, email: canetti@watson.ibm.com; Angelos D. Keromytis, Department of Computer Science, Columbia University, 515 CS Building, 1414 Amsterdam Avenue, New York, NY 10027, email: angelos@cs.columbia.edu.

IKE has a number of deficiencies, the three most important being that the number of rounds is high, that it is vulnerable to denial-of-service (DoS) attacks, and the complexity of the protocol and its specification. This complexity has led to interoperability problems—so much so that, several years after its initial adoption by the IETF, there are still non-interoperating commercial implementations.

While it might be possible to "patch" the IKE protocol to fix some of these problems, it may be perferable to construct a new protocol that more narrowly addresses the requirements "from the ground up." We set out to engineer a new key-exchange protocol specifically for Internet security applications. We call our new protocol "JFK," which stands for "Just Fast Keying."

1.1 Design Goals

We seek a protocol with the following characteristics:

—*Security*: No one other than the participants may have access to the generated key.
—*PFS*: It must approach Perfect Forward Secrecy.
—*Privacy*: It must preserve the privacy of the initiator and/or responder, insofar as possible.
—*Memory-DoS*: It must resist memory exhaustion attacks.
—*Computation-DoS*: It must resist CPU exhaustion attacks on the responder.
—*Availability*: It must protect against easy to mount protocol-specific DoS attacks, for example, in a wireless environment where an attacker can observe everyone's transmissions but cannot interfere with the transmitted packets themselves.
—*Efficiency*: It must be efficient with respect to computation, bandwidth, and number of rounds.
—*Non-negotiated*: It must avoid complex negotiations over capabilities.
—*Simplicity*: The resulting protocol must be as simple as possible, within the constraints of the requirements.

The *Security* requirement is obvious enough (we use the security model of Canetti and Krawczyk [2001; 2002a]). The rest, however, require some discussion.

The *PFS* property is perhaps the most controversial. (PFS is an attribute of encrypted communications allowing for a long-term key to be compromised without affecting the security of past session keys.) Rather than assert that "we must have PFS at all costs," we treat the *amount* of forward secrecy as an engineering parameter that can be traded off against other necessary functions, such as efficiency or resistance to DoS attacks. In fact, this corresponds quite nicely to the reality of today's Internet systems, where a compromise during the existence of a security association (SA) will reveal the plaintext of any ongoing transmissions. Our protocol has a *forward secrecy interval*; SAs are protected against compromises that occur outside of that interval. Specifically, we allow a party to reuse the same secret Diffie–Hellman (DH) exponents for multiple

exchanges within a given time period; this may save a large number of costly modular exponentiations.

The *Privacy* property means that the protocol must not reveal the identity of a participant to any unauthorized party, including an active attacker that attempts to act as the peer. Clearly, it is not possible for a protocol to protect both the initator and the responder against an active attacker; one of the participants must always "go first." In general, we believe that the most appropriate choice is to protect the initator, since the initator is typically a relatively anonymous "client," while the responder's identity may already be known. Conversely, protecting the responder's privacy may not be of much value (except perhaps in peer-to-peer communication): in many cases, the responder is a server with a fixed address or characteristics (e.g., well-known web server). One approach is to allow for a protocol that allows the two parties to negotiate who needs identity protection. In JFK, we decided against this approach: it is unclear what, if any, metric can be used to determine which party should receive identity protection; furthermore, this negotiation could act as a loophole to make initiators reveal their identity first. Instead, we propose two alternative protocols: one that protects the initator against an active attack, and another that protects the responder. These two protocols are respectively named JFKi and JFKr.

The *Memory-DoS* and *Computation-DoS* properties have become more important in the context of recent Internet DoS attacks. Photuris [Karn and Simpson 1999] was the first published key-management protocol for which DoS resistance was a design consideration; we suggest that these properties are at least as important today. We also extend these properties with a general *Availability* requirement, by which we mean that the protocol must try to counter attacks that aim to disrupt a legitimate exchange, especially in environments where an attacker may have limited capabilities in terms of packet modification (e.g., a wireless network).

The *Efficiency* property is worth discussing. In many protocols, key setup must be performed frequently enough that it can become a bottleneck to communication. The key-exchange protocol must minimize computation as well total bandwidth and round trips. Round trips can be an especially important factor when communicating over unreliable media. Using our protocols, only two round trips are needed to set up a working SA. This is a considerable saving in comparison with existing protocols, such as IKE.

The *Nonnegotiated* property is necessary for several reasons. Negotiations create complexity and round trips, and hence should be avoided. DoS resistance is also relevant here; a partially negotiated SA consumes resources.

The *Simplicity* property is motivated by several factors. Efficiency is one; increased likelihood of correctness is another. But our motivation is especially colored by our experience with IKE. Even if the protocol is defined correctly, it must be implemented correctly; as protocols become more complex, implementation and interoperability errors occur more often. This hinders both security and interoperability. Our design follows the traditional design paradigm of successful internetworking protocols: keep individual building blocks as simple as possible; avoid large, complex, monolithic protocols. We have consciously chosen

to omit support for certain features when we felt that adding such support would cause an increase in complexity that was disproportional to the benefit gained.

Protocol design is, to some extent, an engineering activity, and we need to provide for tradeoffs between different types of security. There are tradeoffs that we made during the protocol design, and others, such as that between forward secrecy and computational effort, that are left to the implementation and to the user, for example, selected as parameters during configuration and session negotiation.

2. PROTOCOL DEFINITION

We present two variants of the JFK protocol. Both variants take two round trips (i.e., four messages) and both provide the same level of DoS protection. The first variant, denoted *JFKi,* provides identity protection for the initiator even against active attacks. The identity of the responder is not protected. This type of protection is appropriate for a client–server scenario where the initiator (the client) may wish to protect its identity, whereas the identity of the responder (the server) is public. As discussed in Section 4, this protocol uses the basic design of the ISO 9798-3 key-exchange protocol [IEEE 1993; Canetti and Krawczyk 2001], with modifications that guarantee the properties discussed in Section 1.

The second variant, *JFKr*, provides identity protection for the responder against active adversaries. Furthermore, it protects both sides' identities against passive eavesdroppers. This type of protection is appropriate for a peer-to-peer scenario, where the responder may wish to protect its identity. Note that it is considerably easier to mount active identity-probing attacks against the responder than against the initiator. Furthermore, JFKr provides repudiability on the key exchange, since neither side can prove to a third party that their peer in fact participated in the protocol exchange with them. (In contrast, JFKi authentication is nonrepudiable, since each party signs the other's identity along with session-specific information such as the nonces.) This protocol uses the basic design of the Sign-and-MAC (SIGMA) protocol from Krawczyk [2002], again with the appropriate modifications.

2.1 Notation

First, some notation:

$H_k(M)$ Keyed hash (e.g., HMAC [Krawczyk et al. 1997]) of message M using key k. We assume that H is a pseudorandom function. This also implies that H is a secure message authentication (MAC) function. In some places we make a somewhat stronger assumption relating H and discrete logarithms; see more details within.

$\{M\}_{K_a}^{K_e}$ Encryption using symmetric key K_e, followed by MAC authentication with symmetric key K_a of message M. The MAC is computed over the ciphertext, prefixed with the literal ASCII string `"I"` or `"R"`, depending on who the message sender is (initiator or responder).

$S_x[M]$ Digital signature of message M with the private key belonging to principal x (initiator or responder). It is assumed to be a non-message-recovering signature.

The message components used in JFK are:

- IP_I Initiator's network address.
- g^x DH exponentials; also identifying the group-ID.
- g^i Initiator's current exponential (mod p).
- g^r Responder's current exponential (mod p).
- N_I Initiator nonce, a random bit-string.
- N'_I Initiator's initial nonce, computed as $H(N_I)$.
- N_R Responder nonce, a random bit-string.
- ID_I Initiator's certificates or public-key identifying information.
- ID_R Responder's certificates or public-key identifying information.
- $\mathrm{ID}_{R'}$ An indication by the initiator to the responder as to what authentication information (e.g., certificates) the latter should use.
- HK_R A transient hash key private to the responder.
- sa Cryptographic and service properties of the SA that the initiator wants to establish. It contains a Domain-of-Interpretation, which JFK understands, and an application-specific bit-string.
- sa′ SA information the responder may need to give to the initiator (e.g., the responder's SPI, in IPsec).
- K_{ir} Shared key derived from g^{ir}, N'_I, and N_R used for protecting the application (e.g., the IPsec SA).
- K_e, K_a Shared keys derived from g^{ir}, N'_I, and N_R, used to encrypt and authenticate messages (3) and (4) of the protocol.
- $\mathrm{grpinfo}_R$ All groups supported by the responder, the symmetric algorithms used to protect messages (3) and (4), and the hash function used for key generation.

Both parties must pick a fresh nonce at each invocation of the JFK protocol. The nonces are used in the session-key computation, to provide key independence when one or both parties reuse their DH exponential; the session key will be different between independent runs of the protocol, as long as one of the nonces or exponentials changes. HK_R is a global parameter for the responder—it stays the same between protocol runs, but can change periodically.

2.2 The JFKi Protocol

The JFKi protocol consists of four messages (two round trips):

$$I \to R \ : \ N'_I, g^i, ID_{R'} \tag{1}$$
$$R \to I \ : \ N'_I, N_R, g^r, \mathrm{grpinfo}_R, \mathrm{ID}_R,$$

$$S_R[g^r, \text{grpinfo}_R],$$
$$H_{\text{HK}_R}(g^r, N_R, N'_I, IP_I) \quad (2)$$

$$I \rightarrow R \ : \ N_I, N_R, g^i, g^r,$$
$$H_{\text{HK}_R}(g^r, N_R, N'_I, IP_I),$$
$$\{\text{ID}_I, \text{sa}, S_I[N'_I, N_R, g^i, g^r, \text{ID}_R, \text{sa}]\}_{K_a}^{K_e} \quad (3)$$

$$R \rightarrow I \ : \ \{S_R[N'_I, N_R, g^i, g^r, \text{ID}_I, \text{sa}, \text{sa}'], \text{sa}'\}_{K_a}^{K_e} \quad (4)$$

The keys K_e and K_a, used to protect the confidentiality and integrity of messages (3) and (4), respectively, are computed as $H_{g^{ir}}(N'_I, N_R, \texttt{"1"})$ and $H_{g^{ir}}(N'_I, N_R, \texttt{"2"})$, respectively. The session key, K_{ir}, is $H_{g^{ir}}(N'_I, N_R, \texttt{"0"})$. This key is passed to IPsec or, more generally, to the application that requested key-agreement services. (Note that there may be a difference in the number of bits from the HMAC and the number produced by the raw DH exchange; the 512 least-significant bits are of g^{ir} are used as the key in that case.) If the key used by IPsec is longer than the output of the HMAC, the key extension method of IKE is used to generate more keying material.

Message (1) is straightforward; note that it assumes that the initiator already knows a group and generator that are acceptable to the responder. The initiator can reuse a g^i value in multiple instances of the protocol with the responder, or other responders that accept the same group, for as long as she wishes her forward secrecy interval to be. We discuss how the initiator can discover what groups to use in a later section. This message also contains an indication as to which ID the initiator would like the responder to use to authenticate. $ID_{R'}$ is sent in the clear; however, the responder's ID in message (2) is also sent in the clear, so there is no loss of privacy.

Message (2) is more complex. Assuming that the responder accepts the DH group in the initiator's message (rejections are discussed in Section 2.5), he replies with a signed copy of his own exponential (in the same group, also $\pmod p$), information on what secret key algorithms are acceptable for the next message, a random nonce, his identity (certificates or a string identifying his public key), and an authenticator calculated from a secret, HK_R, known to the responder; the authenticator is computed over the responder's exponential, the two nonces, and the initiator's network address. The responder's exponential may also be reused; again, it is regenerated according to the responder's forward secrecy interval. The signature on the exponential needs to be calculated at the same rate as the responder's forward secrecy interval (when the exponential itself changes). Finally, note that the responder does not need to generate any state at this point, and the only cryptographic operation is a MAC calculation. If the responder is not under heavy load, or if PFS is deemed important, the responder may generate a new exponential and corresponding signature for use in this exchange; of course, this would require keeping some state (the secret part of the responder's DH computation).

Message (3) echoes back the data sent by the responder, including the authenticator. However, instead of the initial nonce, N'_I, the initiator now sends its

source, N_I. The authenticator is used by the responder to verify the authenticity of the returned data. The authenticator also confirms that the sender of the message (3) used the same address as in message (1)—this can be used to detect and counter a "cookie jar" DoS attack.[1] A valid authenticator indicates to the responder that a round trip has been completed (between messages (1), (2), and (3)). To check the authenticator, the responder must first compute N_I' from its source, N_I by applying the hash function H. The purpose of this scheme is to avoid a certain attack in environments where an attacker can eavesdrop traffic but cannot modify already transmitted packets, for example, in a wireless network. In such an environment, if N_I was used instead of N_I', the eavesdropper could construct a valid-looking message (3) (by copying the nonces, exponentials, and the authenticator) that would cause the responder to perform the (expensive) public-key operations, only to then drop the packet because further processing would detect that the message was fake (by failure to decrypt or verify the remainder of the payload, as we shall see shortly). The responder is then left with two options: blacklist the authenticator (causing the exchange with the initiator to fail), or simply discard the packet (thus allowing a computation-based DoS attack). With our approach, however, the eavesdropper cannot produce a valid N_I (since that would imply a weak hash function) and thus cannot produce a message (3) that will pass the authenticator verification phase. If the eavesdropper cannot intercept or preempt a valid message (3), as may be the case with some wireless networks, they cannot hijack the initiator's response to mount this DoS attack. Note that if a legitimate message (3) is transmitted over the wireless network but is somehow not received by the responder, an eavesdropper can use it to mount the previously described attack. Although the scheme is not foolproof, it significantly raises the bar against this DoS attack in certain environments, without hindering the exchange under normal circumstances.

Message (3) also includes the initiator's identity and service request, and a signature computed over the nonces, the responder's identity, and the two exponentials. This latter information is all encrypted and authenticated under keys K_e and K_a, as already described. The encryption and authentication use algorithms specified in grpinfo_R. The responder keeps a copy of recently received messages (3), and their corresponding message (4). Receiving a duplicate (or replayed) message (3) causes the responder to simply retransmit the corresponding message (4), without creating new state or invoking IPsec. This cache of messages can be reset as soon as HK_R is changed. The responder's exponential (g^r) is re-sent by the initiator because the responder may be generating a new g^r for every new JFK protocol run (e.g., if the arrival rate of requests is below some threshold). It is important that the responder deal with repeated messages (3) as described above. Responders that create new state for a repeated message (3) open the door to attacks against the protocol and/or underlying application (IPsec).

[1]The "cookie jar" DoS attack involves an attacker that is willing to reveal the address of one subverted host so as to acquire a valid cookie (or number of cookies) that can then be used by a large number of other subverted hosts to launch a DDoS attack using the valid cookie(s).

Note that the signature is protected by the encryption. This is necessary for identity protection, since everything signed is public except the sa, and that is often guessable. An attacker could verify guesses at identities if the signature were not encrypted.

Message (4) contains application-specific information (such as the responder's IPsec SPI), and a signature on both nonces, both exponentials, and the initiator's identity. Everything is encrypted and authenticated by the same K_e and K_a used in message (3), which are derived from N_I', N_R, and g^{ir}. The encryption and authentication algorithms are specified in grpinfo_R.

2.3 Discussion

The design follows from our requirements. With respect to communication efficiency, observe that the protocol requires only two round trips. The protocol is optimized to protect the responder against DoS attacks on state or computation. The initiator bears the initial computational burden and must establish round-trip communication with the responder before the latter is required to perform expensive operations. At the same time, the protocol is designed to limit the private information revealed by the initiator; she does not reveal her identity until she is sure that only the responder can retrieve it. (An active attacker can replay an old message (2) as a response to the initiator's initial message, but he cannot retrieve the initiator's identity from message (3) because he cannot complete the DH computation.)

The initiator's first message, message (1), is a straightforward DH exponential. Note that this is assumed to be encoded in a self-identifying manner, that is, it contains a tag indicating which modulus and base was used. The nonce N_I' serves three purposes: first, it allows the initiator to reuse the same exponential across different sessions (with the same or different responders, within the initiator's forward secrecy interval) while ensuring that the resulting session key will be different. Secondly, it can be used to differentiate between different parallel sessions (in any case, we assume that the underlying transport protocol, that is, UDP, can handle the demultiplexing by using different ports at the initiator). Lastly, it allows the responder to distinguish between a valid message (3) and one produced by an eavesdropper, as we discussed in the previous section.

Message (2) must require only minimal work for the responder, since at that point he has no idea whether the initiator is a legitimate correspondent or, for example, a forged message from a DoS attack; no round trip has yet occurred with the initiator. Therefore, it is important that the responder not be required at this point to perform expensive calculations or create state. Here, the responder's cost will be a single authentication operation, the cost of which (for HMAC) is dominated by two invocations of a cryptographic hash function, plus generation of a random nonce N_R.

The responder *may* compute a new exponential $g^b \pmod p$ for each interaction. This is an expensive option, however, and at times of high load (or attack) it would be inadvisable. The nonce prevents two successive session keys from being the same, even if both the initiator and the responder are reusing

exponentials. One case when both sides may reuse the same exponentials is when the initiator is a low-power device (e.g., a cellphone) and the responder is a busy server.

A simple way of addressing DoS is to periodically (e.g., once every 30 s) generate an $(r, g^r, H_{\mathrm{HK}_R}(g^r), S_R[g^r])$ tuple and place it in a FIFO queue. As requests arrive (in particular, as valid messages (3) are processed), the first entry from the FIFO is removed; thus, as long as valid requests arrive at under the generation rate, PFS is provided for all exchanges. If the rate of valid protocol requests exceeds the generating rate, a JFK implementation should reuse the last tuple in the FIFO. Notice that in this scheme, the same g^r may be reused in different sessions, if these sessions are interleaved. This does not violate the PFS or other security properties of the protocol.

If the responder is willing to accept the group identified in the initiator's message, his exponential must be in the same group. Otherwise, he may respond with an exponential from any group of his own choosing. The field $\mathrm{grpinfo}_R$ lists what groups the responder finds acceptable, if the initiator should wish to restart the protocol. This provides a simple mechanism for the initiator to discover the groups currently allowed by the responder. That field also lists what encryption and MAC algorithms are acceptable for the next two messages. This is not negotiated; the responder has the right to decide what strength encryption is necessary to use his services.

Note that the responder creates no state when sending this message. If it is fraudulent, that is, if the initiator is nonexistent or intent on perpetrating a DoS attack, the responder will not have committed any storage resources.

In message (3), the initiator echoes content from the responder's message, including the authenticator. The authenticator allows the responder to verify that he is in round-trip communication with a legitimate potential correspondent. By revealing the source, N_I, of the initial nonce, N_I', the initiator denies an eavesdropper the ability to disrupt the protocol exchange by transmitting a valid-looking message (3), as discussed above. The initiator also uses the key derived from the two exponentials and the two nonces to encrypt her identity and service request. The initiator's nonce is used to ensure that this session key is unique, even if both the initiator and the responder are reusing their exponentials and the responder has "forgotten" to change nonces.

Because the initiator can validate the responder's identity before sending her own and because her identifying information (ignoring her public-key signature) is sent encrypted, her privacy is protected from both passive and active attackers. An active attacker can replay an old message (2) as a response to the initiator's initial message, but he cannot retrieve the initiator's identity from message (3) because he cannot complete the DH computation. The service request is encrypted, too, since its disclosure might identify the requester. The responder may wish to require a certain strength of cryptographic algorithm for selected services.

Upon successful receipt and verification of this message, the responder has a shared key with a party known to be the initiator. The responder further knows

what service the initiator is requesting. At this point, he may accept or reject the request.

The responder's processing on receipt of message (3) requires verifying an authenticator and, if that is successful, performing several public-key operations to verify the initiator's signature and certificate chain. The authenticator (requiring three hash operations, two for the HMAC computation and one for deriving N_I' from the received N_I) is sufficient defense against forgery; replays, however, could cause considerable computation. The defense against this is to cache the corresponding message (4); if a duplicate message (3) is seen, the cached response is retransmitted; the responder does not create any new state or notify the application (e.g., IPsec). The key for looking up messages (3) in the cache is the authenticator; this prevents DoS attacks where the attacker randomly modifies the encrypted blocks of a valid message, causing a cache miss and thus more processing to be done at the responder. Further, if the authenticator verifies but there is some problem with the message (e.g., the certificates do not verify), the responder can cache the authenticator along with an indication as to the failure (or the actual rejection message), to avoid unnecessary processing (which may be part of a DoS attack). This cache of messages (3) and authenticators can be purged as soon as HK_R is changed (since the authenticator will no longer pass verification).

Caching message (3) and refraining from creating new state for replayed instances of message (3) also serves another security purpose. If the responder were to create a new state and send a new message (4), and a new sa′ for a replayed message (3), then an attacker who compromised the initiator could replay a recent session with the responder. That is, by replaying message (3) from a recent exchange between the initiator and the responder, the attacker could establish a session with the responder where the session-key would be identical to the key of the previous session (which took place when the initiator was not yet compromised). This could compromise the Forward Security of the initiator.

There is a risk, however, in keeping this message cached for too long: if the responder's machine is compromised during this period, PFS is compromised. We can tune this by changing the MAC key HK_R more frequently. The cache can be reset when a new HK_R is chosen.

In message (4), the responder sends to the initiator any responder-specific application data (e.g., the responder's IPsec SPI), along with a signature on both nonces, both exponentials, and the initiator's identity. All the information is encrypted and authenticated using keys derived from the two nonces, N_I' and N_R, and the DH result. The initiator can verify that the responder is present and participating in the session, by decrypting the message and verifying the enclosed signature.

2.4 The JFKr Protocol

Using the same notation as in JFKi, the JFKr protocol is

$$I \rightarrow R \ : \ N_I', g^i \tag{1}$$

$$R \to I \ : \ N_I', N_R, g^r, \text{grpinfo}_R, H_{\text{HK}_R}\left(g^r, N_R, N_I', IP_I\right) \tag{2}$$

$$I \to R \ : \ N_I, N_R, g^i, g^r, H_{\text{HK}_R}\left(g^r, N_R, N_I', IP_I\right) \{\text{ID}_I, \text{ID}_{R'}, \text{sa}, S_I[N_I', N_R, g^i, g^r, \text{grpinfo}_R]\}_{K_a}^{K_e} \tag{3}$$

$$R \to I \ : \ \{\text{ID}_R, \text{sa}', S_R[g^r, N_R, g^i, N_I']\}_{K_a}^{K_e} \tag{4}$$

As in JFKi, the keys used to protect messages (3) and (4), K_e and K_a, are respectively computed as $H_{g^{ir}}(N_I', N_R, \texttt{"1"})$ and $H_{g^{ir}}(N_I', N_R, \texttt{"2"})$. The session key passed to IPsec (or some other application), K_{ir}, is $H_{g^{ir}}(N_I', N_R, \texttt{"0"})$.

Both parties send their identities encrypted and authenticated under K_e and K_a, respectively, providing both parties with identity protection against passive eavesdroppers. In addition, the party that first reveals its identity is the initiator. This way, the responder is required to reveal its identity only after it verifies the identity of the initiator. This guarantees active identity protection to the responder.

We remark that it is essentially impossible, under current technology assumptions, to have a two-round-trip protocol that provides DoS protection for the responder, passive identity protection for both parties, and active identity protection for the initiator. An informal argument proceeds as follows: If DoS protection is in place, then the responder must be able to send his first message before he computes any shared key. This is so since computing a shared key is a relatively costly operation in current technology. This means that the responder cannot send his identity in the second message, without compromising his identity protection against passive eavesdroppers. This means that the responder's identity must be sent in the fourth (and last) message of the protocol. Consequently, the initiator's identity must be sent before the responder's identity is sent.

2.5 Rejection Messages

Instead of sending messages (2) or (4), the responder can send a "rejection" instead. For message (2), this rejection can only be on the grounds that he does not accept the group that the initiator has used for her exponential. Accordingly, the reply should indicate what groups are acceptable. Since message (2) already contains the field grpinfo_R (which indicates what groups are acceptable), no explicit rejection message is needed. (For efficiency's sake, the group information could also be in the responder's long-lived certificate, which the initiator may already have.)

Message (4) can be a rejection for several reasons, including lack of authorization for the service requested. But it could also be caused by the initiator requesting cryptographic algorithms that the responder regards as inappropriate, given the requester (initiator), the service requested, and possibly other information available to the responder, such as the time of day or the initiator's location as indicated by the network. In these cases, the responder's reply should

list acceptable cryptographic algorithms, if any. The initiator would then send a new message (3), which the responder would accept anew; again, the responder does not create any state until after a successful message (3) receipt.

3. WHAT JFK AVOIDS

By intent, JFK does not do certain things. It is worth enumerating them, if only to stimulate discussion about whether certain protocol features are ever appropriate. In JFK, the "missing" features were omitted by design, in the interests of simplicity.

3.1 Multiple Authentication Options

The most obvious "omission" is any form of authentication other than by certificate chains trusted by the each party. We make no provisions for shared secrets, token-based authentication, certificate discovery, or explicit cross-certification of PKIs. In our view, these are best accomplished by outboard protocols. Initiators that wish to rely on any form of legacy authentication can use the protocols being defined by the IPSRA [Sheffer et al. 2001] or SACRED [Arsenault and Farrell 2001; Gustafson et al. 2001] IETF working groups. While these mechanisms do add extra round trips, the expense can be amortized across many JFK negotiations. Similarly, certificate chain discovery (beyond the minimal capabilities implicit in ID_I and ID_R) should be accomplished by protocols defined for that purpose. By excluding the protocols from JFK, we can exclude them from our security analysis; the only interface between the two is a certificate chain, which by definition is a stand-alone secure object.

We also eliminate negotiation generally, in favor of ukases issued by the responder. The responder is providing a service; it is entitled to set its own requirements for that service. Any cryptographic primitive mentioned by the responder is acceptable; the initiator can choose any it wishes. We thus eliminate complex rules for selecting the "best" choice from two different sets. We also eliminate the need that state be kept by the responder; the initiator can either accept the responder's desires or restart the protocol.

3.2 Phase II and Lack Thereof

JFK rejects the notion of two different phases. As will be discussed in Section 5, the practical benefits of quick mode are limited. Furthermore, we do not agree that frequent rekeying is necessary. If the underlying block cipher is sufficiently limited as to bar long-term use of any one key, the proper solution is to replace that cipher. For example, 3DES is inadequate for protection of very high-speed transmissions, because the probability of collision in CBC mode becomes too high after encryption of 2^{32} plaintext blocks. Using AES instead of 3DES solves that problem without complicating the key exchange.

Phase II of IKE is used for several things; we do not regard any of them as necessary. One is generating the actual keying material used for SAs. It is expected that this will be done several times, to amortize the expense of the Phase I negotiation. A second reason for this is to permit very frequent rekeying. Finally, it permits several separate SAs to be set up, with different parameters.

We do not think these apply. First, with modern ciphers such as AES, there is no need for frequent key changes. AES keys are long enough that brute-force attacks are infeasible. Its longer block size protects against CBC limitations when encrypting many blocks.

We also feel that JFK is efficient enough that avoiding the overhead of a full key exchange is not required. Rather than adding new SAs to an existing Phase I SA, we suggest that a full JFK exchange be initiated instead. We note that the initiator can also choose to reuse its exponential, it if wishes to trade PFS for computation time. If state already exists between the initiator and the responder, they can simply check that the DH exponentials are the same; if so, the result of the previous exponentiation can be reused. As long as one of the two parties uses a fresh nonce in the new protocol exchange, the resulting cryptographic keys will be fresh and not subject to a related key (or other, similar) attack. As we discuss in Section 3.3, a similar performance optimization can be used on the certificate-chain validation.

A second major reason for Phase II is dead-peer detection. IPsec gateways often need to know if the other end of a SA is dead, both to free up resources and to avoid "black holes." In JFK, this is done by noting the time of the last packet received. A peer that wishes to elicit a packet may send a "ping." Such hosts may decline any proposed SAs that do not permit such "ping" packets.

A third reason for Phase II is general SA control, and in particular SA deletion. While such a desire is not wrong, we prefer not to burden the basic key-exchange mechanism with extra complexity. There are a number of possible approaches. Ours requires that JFK endpoints implement the following rule: a new negotiation that specifies an SPD identical to the SPD of an existing SA overwrites it. To some extent, this removes any need to delete an SA if black hole avoidance is the concern; simply negotiate a new SA. To delete an SA without replacing it, negotiate a new SA with a null ciphersuite.

3.3 Rekeying

When a negotiated SA expires (or shortly before it does), the JFK protocol is run again. It is up to the application to select the appropriate SA to use among many valid ones. In the case of IPsec, implementations should switch to using the new SA for outgoing traffic, but would still accept traffic on the old SA (as long as that SA has not expired).

To address performance considerations, we should point out that, properly implemented, rekeying only requires one signature and one verification operation in each direction, if both parties use the same DH exponentials (in which case, the cached result can be reused) and certificates: the receiver of an ID payload compares its hash with those of any cached ID payloads received from the same peer. While this is an implementation detail, a natural location to cache past ID payloads is along with already established SAs (a convenient fact, as rekeying will likely occur before existing SAs are allowed to expire, so the ID information will be readily available). If a match is found and the result has not "expired" yet, then we do not need to revalidate the certificate chain. A previously verified certificate chain is considered valid for the shortest of its CRL

revalidate time, certificate expiration time, OCSP result validity time, and so on. For each certificate chain, there is one such value associated (the time when one of its components becomes invalid or needs to be checked again). Notice that an implementation does not need to cache the actual ID payloads; all that is needed is the hash and the expiration time.

That said, if for some reason fast rekeying is needed for some application domain, it should be done by a separate protocol.

4. DISCUSSION OF THE SECURITY ANALYSIS

Detailed formal definitions and proofs of security of the JFK protocols are beyond the scope of this paper. Nonetheless, below we sketch the main ideas of the analysis, with emphasis on those that require extending the existing techniques for key-agreement protocols.

There are currently two main approaches to analyzing the security of protocols: the formal-methods approach and the cryptographic reduction approach. In the former, the cryptographic components of a protocol are modeled by "ideal boxes." Then automatic theorem-verification tools are applied to the protocol. In this approach, if the verification returns a failure, the protocol has a serious flaw (and the flaw is typically explicitly presented). However, even if the verification procedure does not return a flaw, it still does not follow that the protocol is resistant to all feasible attacks. This is due to the idealized modeling of the cryptographic primitives.

The cryptographic reduction approach provides a less abstract treatment of the security of protocols, taking into account the imperfections of the underlying cryptographic primitives. Specifically, the underlying cryptographic primitives are assumed to be resistant with high probability to all attacks that consume a given amount of resources. The probability and resource amounts are treated as parameters. For this exposition we denote the tuple of parameters the *security level*. A proof of security in this approach derives the security level for the protocol essentially as a function of the security levels of the underlying primitives. Intuitively, a protocol is secure if the security level of the protocol are not much smaller than the security levels of the underlying primitives. As in the formal-methods approach, a proof of security in the cryptographic reduction approach does not imply that the protocol is resistant to all feasible attacks. However, it does imply that any efficient attack on the protocol can be converted into an efficient attack on one of the underlying primitives. Essentially a proof in the cryptographic reduction approach shows that the intractability of attacking the protocol follows if the underlying cryptographic primitives are assumed to be intractable to break. The security of the protocol thus rests solely on the security assumptions on the primitives and not on any implicit or hidden assumptions.

The formal-methods approach, being automatable, has the advantage that it is less susceptible to human errors and oversights in analysis. However, when verifiable proofs are attainable via the cryptographic reduction approach they provide more quantitative information about the relationship between the security of the cryptographic primitives and the security of the protocols. This

quantitative information is often very important when determining the operating parameters of the protocol.

We stress that neither approach provides full analysis of an implementation of the protocol. In particular, issues like coding errors, buffer overflows, and the like are not treated. These should be dealt with carefully, using different tools, on an implementation-by-implementation basis.

Our analysis follows the cryptographic reduction approach. We welcome any additional analysis. In particular, analysis based on formal methods would be a useful complement.

We separate the analysis of the "core security" of the protocol from the analysis of added security features such as DoS protection and identity protection. We discuss first the core security.

4.1 Security of Key-Agreement Protocols

JFK was designed for application environments, such as IPsec, in which many sessions of the same protocol may be active in a given host concurrently. In such a setting the protocol in each host must have a method of multiplexing among various active concurrent sessions. For simplicity, here we assume that each flow has an implicit or explicit session identifier of the initiator's, denoted s_I, which is unique among all active sessions in host I, and that each flow, except perhaps the first, has an implicit or explicit session identifier of the responder's, denoted s_R, which is unique among all active sessions in host R. And we denote the session id s as the pair (s_I, s_R). The cryptographic security issues for concurrent sessions of key-agreement protocols were first formalized by Bellare and Rogaway [1993]. The starting point for our treatment and analysis is based on that of Canetti and Krawczyk [2001], which in turn is based on Bellare and Rogaway [1993]. See these papers for more references and comparisons with other analytical work. We briefly sketch the security model below which we denote the SK-security model [Canetti and Krawczyk 2001].

Very roughly, the core security of a key-exchange protocol boils down to two requirements, *correctness* and *pseudorandomness*, which may be summarized as follows.

Session-Key-Agreement Requirements: If party A generates a key K_A associated with a session-identifier s and peer identity B, and party B generates a key K_B associated with the same session identifier s and peer A, then

(1) *Correctness*: K_A must be equal to K_B.
(2) *Pseudorandomness*: K_A must be indistinguishable from a truly random string to all parties except A, and B, and those parties that have broken into A or B or have compromised that session of A or B.

These requirements must remain true even in the presence of adversaries. There are two essential ideas in modeling the adversary in the SK-security model. The first is to model the network as a star with the adversary at the hub. All messages sent from one host intended for another host first go to the adversary. Furthermore, the adversary can modify, reroute, stall, reorder, inject, and/or drop any and all messages. This captures the adversary's ability to

mount "man-in-the-middle" attacks, as well as eavesdropping on packets before sending them on their way to the intended recipient. The second modification is to allow the adversary to control the timing of events and thus, for example, to control the interleaving of sessions made possible by the concurrent capability of the protocol. In this model the protocol is event/message driven. In particular, a protocol session in a host may be initiated by an *initiate-session* event in the host itself, in which case the host is the initiator in that protocol session. Such an event specifies the intended responder for that session. A protocol session in a host may also be initiated by the receipt of a message from the adversary (i.e., ostensibly from another host), in which case the host is the responder in that protocol session. The adversary is given the power to issue initiate-session events to hosts in any fashion it sees fit. Recall the adversary also controls the timing of the sending and receiving of all messages. The adversary can thus create arbitrary interleavings of messages of multiple sessions within the same host.

As alluded to above, a key-agreement protocol is said to be SK-secure if it meets the security requirements above even in the face of an adversary with the above powers that runs in a feasible amount of time. But to be a bit more precise we need to expand on the pseudorandomness requirement, which is to say we need to describe the meaning of pseudorandomness in our context, and the capabilities given to the adversary in its attempt to distinguish a session key from a truly random string. In addition to dynamically initiating sessions and controlling the timing/content of all received messages, the adversary is allowed to make *session-key queries*. That is, it is allowed to dynamically query a host of its choosing and receive the session key of its choosing of any thus-far completed sessions of that host. (This models compromise of a key by some other protocol that uses the key and perhaps leaks information about it.) The adversary is also allowed to make *session-state queries* and *long-term key queries* (which model different levels of break-ins to the parties). That is, it is allowed to dynamically query and receive the internal information of the session and the long-term keys of the hosts of its choosing. Note that after the adversary has received the long-term keys of a host, all of the session keys subsequently agreed to with that host will be known to the adversary. The adversary must at some point ask for a *challenge query*. That is, it dynamically chooses a host and a thus-far completed session as the *test session*. It receives in response a challenge string that is either the session key of the test session or a random string of the same length with equal probability. The adversary may continue to query and receive session keys or long-term keys after it has made its challenge query. Eventually, the adversary must output an answer that is either "random string" or "session key." The goal of an adversary is for its probability of being correct to exceed 1/2 by as much as possible. The session keys of a protocol meet the pseudorandom requirement as long as the probability that any feasible adversary is correct is at most 1/2 plus a negligible amount.

The alert reader will notice that the adversary can always win the above game with probability 1 unless some additional restrictions are imposed. In particular, the adversary is not allowed to have queried at any time either endpoint host of the test session for the session key of that session or for the

long-term keys of that host. But these are the only restrictions placed on the behavior of the adversary.

Note that a protocol that is secure in this sense is very robust. For example, if an adversary, even one that sees every message between every host, breaks into one session, the rest of the session keys agreed to by the various hosts do not become compromised in any discernible way. In a very strong sense the session keys among all of the sessions behave as independent random strings in terms of computational distinguishability. There is no discernible correlation even between session keys of concurrent sessions in the same host. In fact, as was shown in Canetti and Krawczyk [2002b], this notion of security guarantees strong *secure composability* properties.

We stress that this is only a rough sketch of the requirement. For full details see Canetti and Krawczyk [2001, 2002a].

We first discuss the protocols in the restricted case where the parties do not reuse the private DH exponents for multiple sessions.

4.1.1 *The Core Cryptographic Protocol of JFKi.* The basic cryptographic core of this protocol is the same as the ISO 9798-3 protocol. This protocol can be briefly summarized as below. For subsequent discussion, B plays the role of the initiator and A plays the role of responder. We call this protocol JFKi′

$$A \leftarrow B : N_B, B, g^b \tag{1}$$

$$A \rightarrow B : N_A, N_B, A, g^a, S_A[N_A, N_B, g^a, g^b, B] \tag{2}$$

$$A \leftarrow B : N_A, N_B, S_B[N_A, N_B, g^a, g^b, A] \tag{3}$$

A salient point about this protocol is that each party signs, in addition to the nonces and the two public DH exponents, the identity of the peer. If the peer's identity is not signed then the protocol is completely broken. JFKi inherits the same basic core security. In addition, JFKi adds a preliminary cookie mechanism for DoS protection (which results in adding one flow to the protocol and having the *responder* in JFKi play the role of A), and encrypts the last two messages in order to provide identity protection for the initiator. We will discuss these additions further below.

The session key for JFKi′ is derived from a pseudorandom function H with key g^{ab} and input N_A, N_B and output of the appropriate length for the subsequent session.

The JFKi′ protocol was analyzed and proven secure in Canetti and Krawczyk [2001]. That is, Canetti and Krawczyk [2001] show that the protocol is SK-secure if the signature scheme is secure, the pseudorandom function H is secure, and the decisional Diffie–Hellman (DDH) assumption holds. As we later will need to strengthen this latter assumption we discuss it briefly below.

The DDH problem for the cyclic group G is as follows. With equal probability the input is either a DH tuple (g, g^a, g^b, g^{ab}) or a "random tuple" (g, g^a, g^b, g^r), where g is a generator of G, and where a, b, and r are random in $\{1, 2, \ldots, |G|\}$. The output is a declaration saying that the input is a DH tuple or that the input is a random tuple. The DDH assumption holds for a group G if for all feasible algorithms the probability that the algorithm is correct at most $1/2$ plus a negligible amount.

4.1.2 *The core cryptographic protocol of JFKr.* The basic cryptographic core of JFKr follows the design of the SIGMA protocol [Krawczyk 2002] (which also serves as the basis to the signature mode of IKE). SIGMA was analyzed and proven secure in Canetti and Krawczyk [2002a]. This basic protocol, called JFKr′ for in the ensuing discussion, can be briefly summarized as follows:

$$A \leftarrow B \ : \ N_B, g^b \tag{1}$$

$$A \rightarrow B \ : \ N_A, N_B, A, g^a, S_A[N_A, N_B, g^a, g^b], \\ H_{K_a}(N_A, N_B, A) \tag{2}$$

$$A \leftarrow B \ : \ N_A, N_B, B, S_B[N_A, N_B, g^a, g^b], \\ H_{K_a}(N_A, N_B, B) \tag{3}$$

Here, neither party signs the identity of its peer. Instead, each party includes a MAC applied to its own identity (concatenated with N_A and N_B). The key for the MAC and the session key are derived as in JFKr from a pseudorandom function H with key g^{ab} and inputs N_A, N_B, "2" and N_A, N_B, "0", respectively.

The proof of security of JFKr′ [Canetti and Krawczyk 2002a] assumes that the signature scheme is secure, that H is a secure MAC, that H used for key derivation is a secure pseudorandom function, and that the DDH assumption holds. Note that the function used for the MAC on the wire need not be the same as the function used for key derivation. For protocol simplicity we have assumed they are the same.

In the next section, we discuss the security implications of the encryption of the third and fourth flows in the JFK protocols. In the subsequent section, we define a formal model of identity protection and discuss how it is achieved by virtue of the security of the encryption. Finally, we discuss the security implications of the DoS protection mechanisms employed in the JFK protocols.

4.2 Adding Encryption to the Protocols

In this section, we discuss the security issues associated encrypting and MACing the last two flows. The impetus for adding encryption to the third and fourth flows of the JFK protocols is identity protection which we will discuss more formally in the next section. And proving that these flows actually enjoy the appropriate encryption property is an important step in proving identity protection, as we will discuss below. But the parties may want other information, such as policy information, to remain confidential in these flows as well. Thus, the confidentiality property of these flows is important in its own right.

In both JFKi and JFKr, the third and fourth flows are encrypted and then MACed. The encryption is assumed to be secure against adaptive chosen plaintext attacks (CPA encryption). If the keys for the encryption and the MAC can be shown to be pseudorandom then CPA encryption followed by a secure MAC yields a confidentiality scheme that is secure against adaptive chosen ciphertexts attacks. Thus the confidentiality property of these flows rests on the pseudorandomness of the encryption and MAC keys.

Recall that the keys for the encryption and the MAC are derived by $H_{g^{ir}}(N_I, N_R,$ "1") and $H_{g^{ir}}(N_I, N_R,$ "2") whereas the session key is derived

by $H_{g^{ir}}(N_I, N_R,$ "0"). The pseudorandomness of these values depends on the pseudorandomness of the DH value g^{ir}, which in turn depends on the DDH assumption. It is tempting to conclude that since the session key has been proven to be pseudorandom for protocols essentially identical to JFKi and JFKr except that the encryption and MACing are not present, that the encryption and MAC keys are also pseudorandom given that they are derived in the same manner. It is also tempting to conclude that since the security of the session keys of, say, JFKi′ and JFKr′ are secure that this immediately implies that the session keys of JFKi and JFKr are secure. Unfortunately, given the details of the model described above, this does not follow. The essential reasons are as follows. In the standard SK-security model described above, the adversary only queries for the session key in a host if that session is completed in that host. But the adversary may attempt to break the confidentiality of, say, the third flow without allowing the session to complete. Moreover, in JFKi and JFKr more artifacts of g^{ir} (i.e., the encryptions and MACs using keys derived from g^{ir}) are available to the adversary than in JFKi′ and JFKr′. Intuitively, these artifacts seem to be computationally useless to the adversary. Indeed, they would be if g^{ir} were pseudorandom but this is what we are trying to prove in the first place.

To break this circular reasoning, we must prove something stronger than what is proven in the SK-security model of Canetti and Krawczyk [2001]. We must effectively show that g^{ir} is pseudorandom in a well-defined sense *before* it is used to derive the encryption and MAC keys. Intuitively, this guarantees that the encryptions and MACs thus leak no discernible information about g^{ir} for the remainder of the session. Thus, g^{ir} remains pseudorandom until the end of the session and the hosts can use it to derive a pseudorandom session key. In such a case, we would have the simultaneous pseudorandomness of all of the keys.

More formally, we augment the SK-security model beyond that in Canetti and Krawczyk [2001] as follows. For JFKi and JFKr we allow the adversary to make a *session-symmetric-keys query* for K_a and K_e of a session after either the initiator has sent the third flow or after the responder has received the third flow. We also allow the adversary to ask for a *session-symmetric-keys test* for a host and session of its choosing in which it is presented with equal probability either the K_a and K_e of that session or a random string of the same length. As with the session-symmetric-keys query, the test is either after the initiator has sent the third flow or the responder has received the third flow. After asking for the test, the adversary may continue to make session-symmetric-keys queries, session-key queries, and long-term key queries. The adversary must eventually output an answer that is either "random string" or "session symmetric keys." The goal of an adversary is for its probability of being correct to exceed 1/2 by as much as possible. The encryption and MAC keys of our protocols are pseudorandom as long as the probability that any feasible adversary is correct exceeds 1/2 by only a negligible amount. The only restriction is that the adversary may not make session-symmetric-keys queries or long-term key queries of the "matching" host of the session-symmetric-keys test. This restriction can be described precisely but we omit the details here.

We can prove the simultaneous pseudorandomness of the encryption keys, the MAC keys, and the security of the session keys (i.e., correctness and pseudorandomness) for both JFKi and JFKr. The proof of security assumes that the signature scheme is secure, that H is a secure MAC, that the encryption scheme is secure against adaptive plaintext attacks, that the H used for key derivation is a secure pseudorandom function, and that the DDH assumption holds.

The security of the encryption and MACs of the JFK protocols goes most of the way toward providing identity protection. Nonetheless, there are some additional subtitles that are useful to formalize and we do so below.

Using the security of the encryption and MAC of the JFK protocols the following can be shown to follow. Both JFKi and JFKr provide identity protection against passive adversaries. Furthermore, JFKr provides identity protection against active adversaries for the responder and JFKi provides identity protection against active adversaries for the initiator.

4.3 DoS Protection

There are two elements to the DoS protection in JFKi and JFKr. The first is the cookie in the responder's first reply that enables the responder to not have to store any per session state. The addition of this cookie to the core cryptographic protocols above does not reduce the security. That is, an adversary that breaks the protocol with the cookie can be used to create an adversary that breaks the protocol without the cookie. This reduction is straightforward and the details are omitted here. We thus consider below the variants of JFKi and JFKr, which do not include these cookies.

The second element of DoS protection is to not require the responder to perform any public-key operations in its first reply. To achieve this we do two things. First we add an extra flow at the beginning of the protocol that effectively reverses the roles of A and B in JFKi′ and JFKr′ above. In particular, the responder does not have to produce a fresh signature for each of its first replies as in JFKi′ and JFKr′. Second, we allow the responder (and initiator) to reuse a DH value as often as it sees fit before erasing it and using a new DH value.

The addition of the extra round does not affect the security analysis for the session keys beyond that of, say, JFKi′ and JFKr′ above. However, allowing the reuse of DH values creates two main difficulties. The first requires further changes to the security model above and the second requires a stronger assumption on the cryptographic primitives. We will discuss these issues in turn below.

4.3.1 *PFS and Adaptive Forward Secrecy.* A standard security requirement for a secret key-agreement protocol is PFS [Günther 1989; Diffie et al. 1992a]. Intuitively, PFS guarantees that compromises of a host at time t do not allow an adversary to determine session keys that have been generated and subsequently erased before time t. The models discussed above capture this property, in part. Specifically, even if an adversary sees all of the session keys generated by a host after it chooses a test session in that host, the session key of the test session is still indistinguishable from random for an SK-secure protocol.

In the case when the hosts always use a fresh, random DH value for each session, the models are easily modified to capture all of the PFS requirement. All that is necessary is the partial removal of a restriction placed on the adversary. In the models above, the adversary is not allowed to have made a long-term key query of either of the hosts of the completed test session, either before it made the test session or after. To model PFS, the adversary is allowed to make long-term key queries (in addition to the session-key queries that it is already allowed) of the hosts of a test session *after* the session is complete (but it is still not allowed to have made a long-term key query of either of the hosts of the test session before the test session is begun). A protocol achieves PFS security if it is secure against this more powerful adversary. Assuming that all hosts delete g^{ir} as soon as possible, JFKi and JFKr achieve PFS when they use fresh, random DH values for each session.

We now discuss the security of JFKi and JFKr in the case that hosts reuse their DH exponent and value over multiple KE-sessions within a given time period. This provides some measure of protection against DoS attacks. But it comes at the price of slightly weakening the PFS. To see this suppose that the test session of the adversary involves a responder that was using the DH exponent and value r and g^r, respectively, for that session. Suppose that the responder continues to use these values and the adversary subsequently breaks into this host. In this case the adversary can clearly compromise the test session key.

In spite of this, the JFK protocols with DH reuse still enjoy an "intermediate" level of forward secrecy, which we call *adaptive forward secrecy* (AFS). The intuition is as follows. We consider that the DH exponent and value are used in phases by a host. During a phase, a single DH exponent and value are used for all sessions that are initiated in that phase. A host may choose new DH values and begin a new phase at any time, and it may make that decision to start a new phase in an adaptive manner. (We assume here that when a new phase starts, the DH values of the previous stage are erased and any received messages generated by sessions started with previous DH values receive a failure reply. We can also model protocols that keep several DH values in memory for received messages but for simplicity we do not consider that here.)

Clearly, with such a protocol, if an adversary breaks into a host even at the end of a phase, all of the session keys generated during that phase would be compromised. But some protocols may have the property that even if a host is broken into during a phase, all session keys generated during all previous phases remain secure. Protocols with this property are said to have AFS.

To formalize AFS, we modify the basic definition of SK-security yet again (this modification comes instead of the modification leading to the definition of PFS). The modification is as follows.

On top of its usual capabilities in the SK model, we provide the adversary with a new capability: At any time during the computation, the adversary can activate a new AFS phase in a host of its choosing. In response, the protocol may instruct the host to perform some protocol steps.

The distinguishing game for the adversary is the same as in the first SK security definition but with a new restriction on the adversary's long-term key

queries. There cannot be a long-term key query of either host of the test session at any time during either AFS phase in which the test session was initiated and completed.

A natural four round variant of JFKr and JFKi without the encryption and MACing for identity protection but with DH reuse can be shown to have AFS security under the DDH assumption. However, as we will discuss below, the DDH assumption is not sufficient for AFS security or identity protection when the third and fourth flows of the JFK protocols are encrypted and MACed.

4.3.2 *The Malleability of DH Values.* Reusing the DH values for multiple sessions may potentially cause another problem, which is unrelated to forward secrecy, and has to do with the "core security" of the protocol. Let us sketch this potential problem and the way we deal with it in the analysis.

Consider the following scenario in JFKi. The adversary starts a new AFS phase in hosts I and R and records an entire session-key protocol, where I and R play the role of the initiator and responder, respectively. The adversary subsequently gives an initiate-session event to I with responder R. The adversary receives the first flow from I but does not pass it on to R. Instead, it sends back to I a replay of R's previous flow 2 but with I's new nonce and a new DH value, say, g^{2r}. Note that the adversary can easily compute this given just the g^r it saw in the first complete session. I receives this message, calculates new encryption and MAC keys using H with function key g^{2ir}, and sends flow 3 to the adversary.

This is precisely where a problem occurs with the standard security assumptions of our the cryptographic primitives. The security of a pseudorandom function such as H assumes that the function keys used by H are random and independent. But here the adversary has induced I to compute H using a pair of function keys that are random but not independent. In fact, the dependence is set by the adversary. This is an example of the *malleability* of the DH key exchange. In this example one function key is the square of the other.

To achieve a proof of AFS security and identity protection when DH values are reused in the JFK protocols we require stronger assumptions on our cryptographic primitives. Rather than using the standard pseudorandom function security assumption for H and the standard DDH assumption, we use what we call the combined H/DDH security assumption. Roughly, this assumption is as follows. An adversary is given g, g^r, and g^i, for random i and r, and in addition is given $H_{g^{ir}}(x)$ for a known x potentially of its choosing. Combined H/DDH security requires that this value remains computationally indistinguishable from a random value even when the adversary queries for and is given the values of $H_k(y)$ for y's of its choosing and k's of the form u^r or v^i, for u or v of its choosing in G. The only restriction is that it may not choose $y = x$ and $u = g^i$ or $y = x$ and $v = g^r$ since in these cases the query would return the value $H_{g^{ir}}(x)$ itself.

As an example, since the adversary knows g^i, it can compute a u of the form g^{ik} for a k of its choosing. It can subsequently query H with key g^{ikr}. While the exact value of the two keys remains unknown to the adversary, it does know that the second key is a kth power of the original DH key.

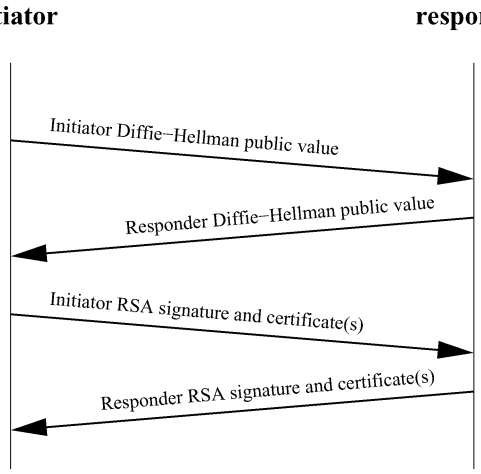

Fig. 1. Four-message StS key-agreement protocol.

Using this combined H/DDH assumption, in concert with secure signatures, encryption, and MACS, we can show that the JFK protocols with DH reuse achieve AFS security and identity protection.

5. RELATED WORK

The basis for most key-agreement protocols based on public-key signatures has been the Station-to-Station (StS) [Diffie et al. 1992a] protocol. In its simplest form, shown in Figure 1, this consists of a DH exchange, followed by a public-key signature authentication step, typically using the RSA algorithm in conjunction with some certificate scheme such as X.509. In most implementations, the second message is used to piggy-back the responder's authentication information, resulting in a three-message protocol, shown in Figure 2. Other forms of authentication may be used instead of public-key signatures (e.g., Kerberos[Miller et al. 1987] tickets, or preshared secrets), but these are typically applicable in more constrained environments. While the short version of the protocol has been proven to be the most efficient [Gong 1995] in terms of messages and computation, it suffers from some obvious DoS vulnerabilities.

5.1 Internet Key Exchange (IKE)

The Internet key-exchange (IKE) protocol [Harkins and Carrel 1998] is the current IETF standard for key establishment and SA parameter negotiation. IKE is based on the ISAKMP [Maughan et al. 1998] framework, which provides encoding and processing rules for a set of payloads commonly used by security protocols, and the Oakley protocol, which describes an adaptation of the StS protocol for use with IPsec.[2] The public-key encryption modes of IKE are based on SKEME [Krawczyk 1996].

[2]We remark, however, that the actual cryptographic core of IKE's signature mode is somewhat different than Oakley. In Oakley the peer authentication is guaranteed by having each party explicitly

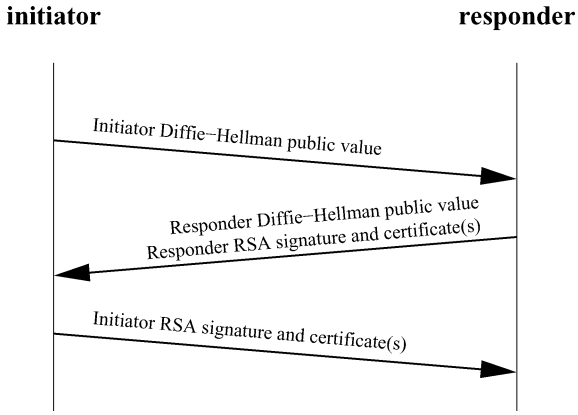

Fig. 2. Three-message StS key-agreement protocol.

IKE is a two-phase protocol: during the first phase, a secure channel between the two key-management daemons is established. Parameters such as an authentication method, encryption/hash algorithms, and a DH group are negotiated at this point. This set of parameters is called a "Phase I SA." Using this information, the peers authenticate each other and compute key material using the DH algorithm. Authentication can be based on public-key signatures, public-key encryption, or preshared passphrases. There are efforts to extend this to support Kerberos tickets [Miller et al. 1987] and handheld authenticators. It should also be noted that IKE can support other key establishment mechanisms (besides DH), although none has been proposed yet.[3]

Furthermore, there are two variations of the Phase I message exchange, called "main mode" and "aggressive mode." Main mode provides identity protection, by transmitting the identities of the peers encrypted, at the cost of three message round trips (see Figure 3). Aggressive mode provides somewhat weaker guarantees, but requires only three messages (see Figure 4).

As a result, aggressive mode is very susceptible to untraceable[4] DoS attacks against both computational and memory resources [Simpson 1999]. Main mode is also susceptible to untraceable memory exhaustion DoS attacks, which must be compensated for in the implementation using heuristics for detection and avoidance. To wit:

—The responder has to create state upon receiving the first message from the initiator, since the Phase I SA information is exchanged at that point. This allows for a DoS attack on the responder's memory, using random source-IP addresses to send a flood of requests. To counter this, the responder could

sign the peer identity. In contrast, IKE guarantees peer authentication by having each party MAC *its own* identity using a key derived from the agreed DH secret. This method of peer authentication is based on the SIGMA design [Krawczyk 2002].

[3]There is ongoing work (still in its early stages) in the IETF to use IKE as a transport mechanism for Kerberos tickets, for use in protecting IPsec traffic.

[4]The attacker can use a forged address when sending the first message in the exchange.

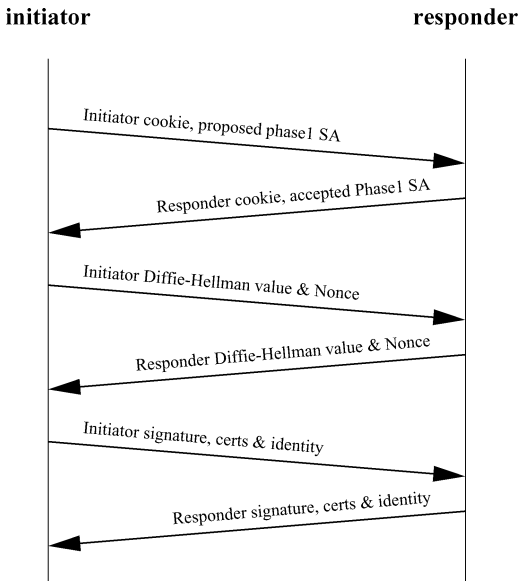

Fig. 3. IKE main mode exchange with certificates.

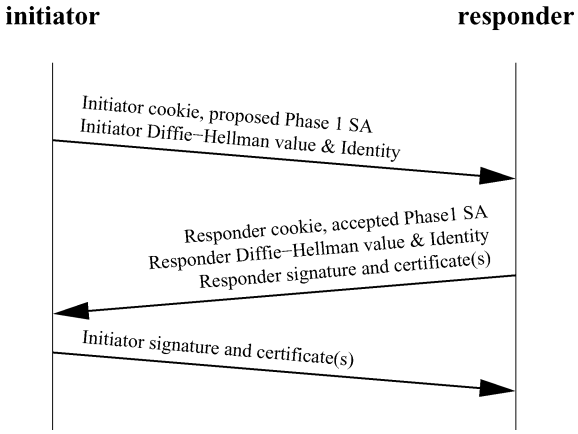

Fig. 4. IKE aggressive mode exchange with certificates.

employ mechanisms similar to those employed in countering TCP SYN attacks [Heberlein and Bishop 1996; CERT 1996; Schuba et al. 1997]. JFK maintains no state at all after receiving the first message.

—An initiator who is willing to go through the first message round trip (and thus identify her address) can cause the responder to do a DH exponential generation as well as the secret key computation on reception of the third message of the protocol. The initiator could do the same with the fifth message of the protocol, by including a large number of bogus certificates, if the responder blindly verifies all signatures. JFK mitigates the effects of this attack by reusing the same exponential across different sessions.

The second phase of the IKE protocol is commonly called "quick mode" and results in IPsec SAs being established between the two negotiating parties, through a three-message exchange. Parameters such as the IP security protocol to use (ESP/AH), security algorithms, the type of traffic that will be protected, and so on, are negotiated at this stage. Since the two parties have authenticated each other and established a shared key during Phase I, quick mode messages are encrypted and authenticated using that information. Furthermore, it is possible to derive the IPsec SA keying material from the shared key established during the Phase I DH exchange. To the extent that multiple IPsec SAs between the same two hosts are needed, this two-phase approach results in faster and more lightweight negotiations (since the same authentication information and keying material is reused).

Unfortunately, two hosts typically establish SAs protecting all the traffic between them, limiting the benefits of the two-phase protocol to lightweight rekeying. If PFS is desired, this benefit is further diluted.

Another problem of the two-phase nature of IKE manifests itself when IPsec is used for fine-grained access control to network services. In such a mode, credentials exchanged in the IKE protocol are used to authorize users when connecting to specific services. Here, a complete Phase I & Phase II exchange will have to be done for each connection (or, more generally, traffic class) to be protected, since credentials, such as public-key certificates, are only exchanged during Phase I.

IKE protects the identities of the initiator and responder from eavesdroppers.[5] The identities include public keys, certificates, and other information that would allow an eavesdropper to determine which principals are trying to communicate. These identities can be independent of the IP addresses of the IKE daemons that are negotiating (e.g., temporary addresses acquired via DHCP, public workstations with smartcard dongles, and so on). However, since the initiator reveals her identity first (in message (5) of Main Mode), an attacker can pose as the responder until that point in the protocol. The attackers cannot complete the protocol (since they do not possess the responder's private key), but they can determine the initiator's identity. This attack is not possible on the responder, since she can verify the identity of the initiator before revealing her identity (in message (6) of Main Mode). However, since most responders would correspond to servers (firewalls, web servers, and so on), the identity protection provided to them seems not as useful as protecting the initiator's identity.[6] Fixing the protocol to provide identity protection for the initiator would involve reducing it to five messages and having the responder send the contents of message (6) in message (4), with the positive side-effect of reducing the number of messages, but breaking the message symmetry and protocol modularity.

Finally, thanks to the desire to support multiple authentication mechanisms and different modes of operation (Aggressive versus Main mode, Phase I/II distinction), both the protocol specification and the implementations tend to

[5]Identity protection is provided only in Main Mode (also known as Identity Protection Mode); Aggressive Mode does not provide identity protection for the initiator.

[6]One case where protecting the responder's identity can be more useful is in peer-to-peer scenarios.

be bulky and fairly complicated. These are undesirable properties for a critical component of the IPsec architecture.

Several works, including Ferguson and Schneier [1999], Kaufman and Perlman [2000], and Kaufman et al. [2001], point out many deficiencies in the IKE protocol, specification, and common implementations. They suggest removing several features of the protocol (e.g., aggressive mode, public-key encryption mode, and so on), restore the idea of stateless cookies, and protect the initiator's (instead of the responder's) identity from an active attacker. They also suggest some other features, such as one-way authentication (similar to what is common practice when using SSL/TLS [Dierks and Allen 1999] on the web). These major modifications would bring the IKE protocol closer to JFK, although they would not completely address the DoS issues.

A measure of the complexity of IKE can be found in the analyses done in Meadows Meadows [1999a, 2000]. No less than 13 different subprotocols are identified in IKE, making understanding, implementation, and analysis of IKE challenging. While the analysis did not reveal any attacks that would compromise the security of the protocol, it did identify various potential attacks (DoS and otherwise) that are possible under some *valid* interpretations of the specification and implementation decisions.

Some work has been done towards addressing, or at least examining, the DoS problems found in IKE Matsuura and Imai [1999, 2000] and, more generally, in public-key-authentication protocols [Leiwo et al. 2000; Jakobsson and Juels 1999]. Various recommendations on protocol design include use of client puzzles [Juels and Brainard 1999; Aura et al. 2000], stateless cookies [Oppliger 1999], forcing clients to store server state, rearranging the order of computations in a protocol [Hirose and Matsuura 1999], and the use of a formal-methods framework for analyzing the properties of protocols with respect to DoS attacks [Meadows 1999b]. The advantages of being stateless, at least in the beginning of a protocol run, were recognized in the security protocol context in Janson et al. [1997] and Aura and Nikander [1997]. The latter presented a three-message version of IKE, similar to JFK, that did not provide the same level of DoS protection as JFK does, and had no identity protection.

5.2 IKEv2

IKEv2 [Harkins et al. 2002] is another proposal for replacing the original IKE protocol. The cryptographic core of the protocol, as shown in Figure 5, is very similar to JFKr. The main differences between IKEv2 and JFKr are:

—IKEv2 implements DoS protection by optionally allowing the responder to respond to a message (1) with a cookie, which the sender has to include in a new message (1). Under normal conditions, the exchange would consist of the four messages shown; however, if the responder detects a DoS attack, it can start requiring the extra round trip. One claimed benefit of this extra round trip is the ability to avoid memory-based DoS

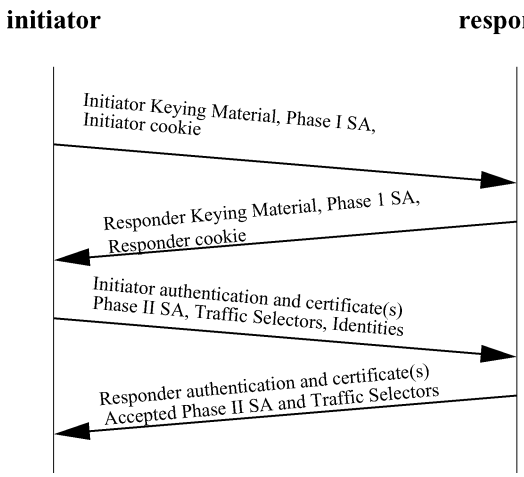

Fig. 5. IKEv2 protocol exchange.

attacks against the fragmentation/reassembly part of the networking stack. (Briefly, the idea behind such an attack is that an attacker can send many incomplete fragments that fill out the reassembly queue of the responder, denying service to other legitimate initiators. In IKEv2, because the "large" messages are the last two in the exchange, it is possible for the implementation to instruct the operating system to place fragments received from peers that completed a round trip to a separate, reserved reassembly queue.)

—IKEv2 supports a Phase II exchange, similar to the Phase I/Phase II separation in the original IKE protocol. It supports creating subsequent IPsec SAs with a single round trip, as well as SA-teardown using this Phase II.

—IKEv2 proposals contain multiple options that can be combined in arbitrary ways; JFK, in contrast, takes the approach of using ciphersuites, similar to the SSL/TLS protocols [Dierks and Allen 1999].

—IKEv2 supports legacy authentication mechanisms (in particular, preshared keys). JFK does not, by design, support other authentication mechanisms, as discussed in Section 3; while it is easy to do so (and we have a variant of JFKr that can do this without loss of security), we feel that the added value compared to the incurred complexity does not justify the inclusion of this feature in JFK.

Apart from these main differences, there are a number of superficial ones (e.g., the "wire" format) that are more a matter of taste than of difference in protocol design philosophy. The authors of the two proposals have helped create a joint draft [Hoffman 2002], submitted to the IETF IPsec Working Group. In that draft, a set of design options reflecting the differences in the two protocols is presented to the working group. Concurrent with the writing of this paper, and based on this draft, a unified proposal is being written. This unified proposal combines properties from both JFK and IKEv2. It adopts the approach of setting up a SA within two round trips, while providing DoS protection for the

responder (and, in particular, allowing the responder to be almost completely stateless between the sending of message (2) and the receipt of message (3).)

5.3 Other Protocols

The predecessor to IKE, Photuris [Karn and Simpson 1999], first introduced the concept of cookies to counter "blind" DoS attacks. The protocol itself is a six-message variation of the StS protocol. It is similar to IKE in the message layout and purpose, except that the SA information has been moved to the third message. For rekeying, a two-message exchange can be used to request a unidirectional SPI (thus, to completely rekey, four messages are needed). Photuris is vulnerable to the same computation-based DoS attack as IKE, mentioned above. Nonetheless, one of the variants of this protocol has four messages and provided DoS protection via stateless cookies.

SKEME [Krawczyk 1996] shares many of the requirements for JFK, and many aspects of its design were adopted in IKE. It serves more as a set of protocol building blocks, rather than a specific protocol instance. Depending on the specific requirements for the key-management protocol, these blocks could be combined in several ways. An interesting aspect of SKEME is its avoidance of digital signatures; public-key encryption is used instead, to provide authentication assurances. The reason behind this was to allow both parties of the protocol to be able to repudiate the exchange.

SKIP [Aziz and Patterson 1995] was an early proposal for an IPsec key-management mechanism. It uses long-term DH public keys to derive long-term shared keys between parties, which is used to distribute session keys between the two parties. The distribution of the session key occurs in-band, that is, the session key is encrypted with the long-term key and is injected in the encrypted packet header. While this scheme has good synchronization properties in terms of rekeying, the base version lacks any provision for PFS. It was later provided via an extension [Aziz 1996]. However, as the authors admit, this extension detracts from the original properties of SKIP. Furthermore, there is no identity protection provided, since the certificates used to verify the DH public keys are (by design) publicly available, and the source/destination master identities are contained in each packet (so that a receiver can retrieve the sender's DH certificate). The latter can be used to mount a DoS attack on a receiver, by forcing them to retrieve and verify a DH certificate, and then compute the DH shared secret.

The host identity payload (HIP) [Moskowitz 2001] uses cryptographic public keys as the host identifiers, and introduces a set of protocols for establishing SAs for use in IPsec. The HIP protocol is a four-packet exchange, and uses client puzzles to limit the number of sessions an attacker can initiate. HIP also allows for reuse of the DH value over a period of time, to handle a high rate of sessions. For rekeying, a HIP packet protected by an existing IPsec session is used. HIP does not provide identity protection, and it depends on the existence of an out-of-band mechanism for distributing keys and certificates, or on extra HIP messages for exchanging this information (thus, the message count is effectively 6, or even 8, for most common usage scenarios).

6. CONCLUSION

Over the years, many different key-exchange protocols have been proposed. Some have had security flaws; others have not met certain requirements.

JFK addresses the first issue by simplicity, and by a proof of correctness. (Again, full details of this are deferred to the analysis paper.) We submit that proof techniques have advanced enough that new protocols should not be deployed without such an analysis. We also note that the details of the JFK protocol changed in order to accommodate the proof: tossing a protocol over the wall to the theoreticians is not a recipe for success. But even a proof of correctness is not a substitute for simplicity of design; apart from the chance of errors in the formal analysis, a complex protocol implies a complex implementation, with all the attendant issues of buggy code and interoperability problems.

The requirements issue is less tractable because it is not possible to foresee how threat models or operational needs will change over time. Thus, StS is not suitable for an environment where DoS attacks are a concern. Another comparatively recent requirement is identity protection. But the precise need—whose identity should be protected, and under what threat model—is still unclear, hence the need for both JFKi and JFKr.

Finally, and perhaps most important, we show that some attributes often touted as necessities are, in fact, susceptible to a cost–benefit analysis. Everyone understands that cryptographic primitives are not arbitrarily strong, and that cost considerations are often used in deciding on algorithms, key lengths, block sizes, and so on. We show that DoS-resistance and PFS have similar characteristics, and that it is possible to improve some aspects of a protocol (most notably the number of round trips required) by treating others as parameters of the system, rather than as absolutes.

ACKNOWLEDGMENTS

Ran Atkinson, Matt Crawford, Paul Hoffman, and Eric Rescorla provided useful comments, and discussions with Hugo Krawczyk proved very useful. Dan Harkins suggested the inclusion of IP_I in the authenticator. David Wagner made useful suggestions on the format of message (2) in JFKi. The design of the JFKr protocol was influenced by the SIGMA and IKEv2 protocols. Yogesh Swami proposed the scheme protecting against message (3) DoS attacks.

REFERENCES

AIELLO, W., BELLOVIN, S. M., BLAZE, M., CANETTI, R., IOANNIDIS, J., KEROMYTIS, A. D., AND REINGOLD, O. 2003. Efficient, DoS-resistant, secure key exchange for internet protocols. In *Proceedings of the ACM Computer and Communications Security (CCS) Conference*. 48–58.

ARSENAULT, A. AND FARRELL, S. 2001. Securely available credentials—requirements. Request for Comments 3157, Internet Engineering Task Force (Aug.).

AURA, T. AND NIKANDER, P. 1997. Stateless connections. In *Proceedings of the International Conference on Information and Communications Security (ICICS '97)*, Lecture Notes in Computer Science, vol. 1334. Springer, Berlin, 87–97.

AURA, T., NIKANDER, P., AND LEIWO, J. 2000. DOS-resistant authentication with client puzzles. In *Proceedings of the Eighth International Workshop on Security Protocols*.

AZIZ, A. 1996. SKIP extension for perfect forward secrecy (PFS). Internet Draft, Internet Engineering Task Force (Aug.).

AZIZ, A. AND PATTERSON, M. 1995. Simple key management for internet protocols (SKIP). In *Proceedings of the 1995 INET Conference*.

BELLARE, M. AND ROGAWAY, P. 1993. Entity authentication and key distribution. In *Proceedings of the Crypto Conference*.

CANETTI, R. AND KRAWCZYK, H. 2001. Analysis of key-exchange protocols and their use for building secure channels. In *Proceedings of the Eurocrypt Conference*.

CANETTI, R. AND KRAWCZYK, H. 2002a. Security analysis of IKE's signature-based key-exchange protocol. In *Proceedings of the Crypto Conference*.

CANETTI, R. AND KRAWCZYK, H. 2002b. Universally composable notions of key exchange and secure channels. In *Proceedings of the EuroCrypt Conference*.

CERT. 1996. *Advisory CA-96.21: TCP SYN Flooding*. Available at ftp://info.cert.org/pub/cert_advisories/CA-96.21.tcp_syn_flooding.

DIERKS, T. AND ALLEN, C. 1999. The TLS Protocol Version 1.0. Request for Comments (Proposed Standard) 2246, Internet Engineering Task Force (Jan.).

DIFFIE, W., VAN OORSCHOT, P., AND WIENER, M. 1992a. Authentication and authenticated key exchanges. *Des. Codes Cryptogr. 2*, 2, 107–125.

FERGUSON, N. AND SCHNEIER, B. 1999. *A Cryptographic Evaluation of IPSec*. Available at `http://www.counterpane.com/ipsec.html`.

GONG, L. 1995. Efficient network authentication protocols: lower bounds and optimal implementations. *Distrib. Comput. 9*, 3, 131–145.

GÜNTHER, C. G. 1989. An identity-based key-exchange protocol. In *Proceedings of the EuroCrypt Conference*. 29–37.

GUSTAFSON, D., JUST, M., AND NYSTROM, M. 2001. Securely available credentials—Credential server framework. Internet Draft, Internet Engineering Task Force (Aug.). Work in progress.

HARKINS, D. AND CARREL, D. 1998. The internet key exchange (IKE). Request for Comments (Proposed Standard) 2409, Internet Engineering Task Force (Nov.).

HARKINS, D., KAUFMAN, C., KENT, S., KIVINEN, T., AND PERLMAN, R. 2002. Proposal for the IKEv2 protocol. Internet Draft, Internet Engineering Task Force (April). Work in progress.

HEBERLEIN, L. AND BISHOP, M. 1996. Attack class: Address spoofing. In *Proceedings of the 19th National Information Systems Security Conference*. 371–377.

HIROSE, S. AND MATSUURA, K. 1999. Enhancing the resistance of a provably secure key agreement protocol to a denial-of-service attack. In *Proceedings Second International Conference on Information and Communication Security (ICICS '99)*. 169–182.

HOFFMAN, P. 2002. Features of proposed successors to IKE. Internet Draft, Internet Engineering Task Force (April). Work in progress.

IEEE. 1993. *Entity Authentication Mechanisms—Part 3: Entity Authentication Using Asymmetric Techniques*. Tech. Rep. ISO/IEC IS 9798-3, ISO/IEC.

JAKOBSSON, M. AND JUELS, A. 1999. Proofs of work and bread pudding protocols. In *Proceedings of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security*.

JANSON, P., TSUDIK, G., AND YUNG, M. 1997. Scalability and flexibility in authentication services: The KryptoKnight approach. In *Proceedings of the IEEE INFOCOM*. 725–736.

JUELS, A. AND BRAINARD, J. 1999. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS '99)*. 151–165.

KARN, P. AND SIMPSON, W. 1999. Photuris: Session-key management protocol. Request for Comments 2522, Internet Engineering Task Force (Mar.).

KAUFMAN, C. ET AL. 2001. Code-preserving Simplifications and Improvements to IKE. Internet Draft, Internet Engineering Task Force (July). Work in progress.

KAUFMAN, C. AND PERLMAN, R. 2000. Analysis of IKE. In *IEEE Trans. Netw. Comput.* (Nov.).

KRAWCZYK, H. 1996. SKEME: A versatile secure key exchange mechanism for internet. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*.

KRAWCZYK, H. 2002. Invited Talk. SIGMA: the SIGn-and-MAc approach to authenticated Diffie–Hellman and its use in the IKE protocols. In *Proceedings of the Crypto Conference*.

KRAWCZYK, H., BELLARE, M., AND CANETTI, R. 1997. HMAC: keyed-hashing for message authentication. Request for Comments 2104, Internet Engineering Task Force (Feb.).

LEIWO, J., NIKANDER, P., AND AURA, T. 2000. Towards network denial of service resistant protocols. In *Proceedings of the 15th International Information Security Conference (IFIP/SEC)*.

MATSUURA, K. AND IMAI, H. 1999. Resolution of ISAKMP/Oakley key-agreement protocol resistant against denial-of-service attack. In *Proceedings of Internet Workshop (IWS '99)*. 17–24.

MATSUURA, K. AND IMAI, H. 2000. Modified aggressive mode of Internet key exchange resistant against denial-of-service attacks. *IEICE Trans. Inf. Syst. E83-D*, 5 (May), 972–979.

MAUGHAN, D., SCHERTLER, M., SCHNEIDER, M., AND TURNER, J. 1998. Internet security association and key management protocol (ISAKMP). Request for Comments (Proposed Standard) 2408, Internet Engineering Task Force (Nov.).

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

MEADOWS, C. 1999b. A formal framework and evaluation method for network denial of service. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. 4–13.

MEADOWS, C. 2000. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX 2000)*. IEEE Computer Society Press, 237–250.

MILLER, S. P., NEUMAN, B. C., SCHILLER, J. I., AND SALTZER, J. H. 1987. *Kerberos Authentication and Authorization System*. Tech. Rep., MIT (Dec.).

MOSKOWITZ, R. 2001. The Host Identity Payload. Internet Draft, Internet Engineering Task Force (July). Work in progress.

OPPLIGER, R. 1999. Protecting key exchange and management protocols against resource clogging attacks. In *Proceedings of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security (CMS '99)*. 163–175.

SCHUBA, C., KRSUL, I., KUHN, M., SPAFFORD, E., SUNDARAM, A., AND ZAMBONI, D. 1997. Analysis of a denial of service attack on TCP. In *IEEE Security and Privacy Conference*, Oakland. 208–223.

SHEFFER, Y., KRAWCZYK, H., AND ABOBA, B. 2001. PIC, a pre-IKE credential provisioning protocol. Internet Draft, Internet Engineering Task Force (Nov.). Work in progress.

SIMPSON, W. A. 1999. IKE/ISAKMP Considered Harmful. *USENIX; login:* (Dec.).

Received July 2003; revised March 2004; accepted March 2004