Efficient, DoS-Resistant, Secure Key Exchange for Internet Protocols*

William Aiello
AT&T Labs Research
aiello@research.att.com

Steven M. Bellovin
AT&T Labs Research
smb@research.att.com

Matt Blaze
AT&T Labs Research
mab@research.att.com

Ran Canetti
IBM T.J. Watson Research Center
canetti@watson.ibm.com

John Ioannidis
AT&T Labs Research
ji@research.att.com

Angelos D. Keromytis
Columbia University
angelos@cs.columbia.edu

Omer Reingold
AT&T Labs Research
omer@research.att.com

Categories and Subject Descriptors

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

General Terms

Security, Reliability, Standardization

Keywords

Cryptography, Denial of Service Attacks

ABSTRACT

We describe 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 trade-offs, most notably the ability to balance the need for perfect forward secrecy against susceptibility to 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, IKE[15], is the current standard. 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 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.)

*This work was partially supported by DARPA under Contract F39502-99-1-0512-MOD P0001.

CCS'02, November 18-22, 2002, Washington, DC USA

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 narrorwly 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.

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 [7, 8]). 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 perfect forward secrecy 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 denial-of-service 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 will reveal the plaintext of any ongoing transmissions. Our protocol has a *forward secrecy interval*; security associations are protected against compromises that occur outside of that interval. Specifically, we allow a party to reuse the same secret Diffie-Hellman 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.

The *Memory-DoS* and *Computation-DoS* properties have become more important in the context of recent Internet denial-of-service attacks. Photuris[24] 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.

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 security association. This is a considerable saving in comparison with existing protocols, such as IKE.

The *Non-Negotiated* property is necessary for several reasons. Negotiations create complexity and round trips, and hence should be avoided. Denial of service resistance is also relevant here; a partially-negotiated security association 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 trade-offs between different types of security. There are trade-offs 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, *e.g.,* 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 [20, 7], with modifications that guarantee the properties discussed in the Introduction.

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 non-repudiable, 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 [28], again with the appropriate modifications.

2.1 Notation

First, some notation:

- $H_k(M)$ Keyed hash (*e.g.,* HMAC[29]) 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.
- $\{M\}^{K_e}_{K_a}$ 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 Diffie-Hellman (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_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 security association (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.

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):

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

$$R \rightarrow I \quad : \quad N_I, N_R, g^r, \text{grpinfo}_R, \text{ID}_R, S_R[g^r, \text{grpinfo}_R], H_{\text{HK}_R}(g^r, N_R, N_I, IP_I) \tag{2}$$

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

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

The keys used to protect Messages (3) and (4), K_e and K_a, 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 passed to IPsec (or some other application), K_{ir}, is $H_{g^{ir}}(N_I, N_R, \texttt{"0"})$. (Note that there may be a difference in the number of bits from the HMAC and the number produced by the raw Diffie-Hellman 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 Diffie-Hellman 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 Diffie-Hellman computation).

Message (3) echoes back the data sent by the responder, including the authenticator. 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 roundtrip has been completed (between Messages (1), (2), and (3)). The message 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).

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 denial of service attacks on state or computation. The initia-

[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).

tor 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 Diffie-Hellman computation).

The initiator's first message, Message (1), is a straightforward Diffie-Hellman exponential. Note that this is assumed to be encoded in a self-identifying manner, *i.e.,* it contains a tag indicating which modulus and base was used. The nonce N_I serves two 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, *i.e.,* UDP, can handle the demultiplexing by using different ports at the initiator).

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, *e.g.,* a forged message from a denial of service 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 seconds) 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 non-existent or intent on perpetrating a denial-of-service 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. 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 Diffie-Hellman 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 (again requiring two hash operations) 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, perfect forward secrecy 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 Diffie-Hellman 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:

$$I \to R: \quad N_I, g^i \tag{1}$$

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

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

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

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[41] or SACRED[1, 14] 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 security associations. 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 security associations 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 perfect forward secrecy for computation time. If state already exists between the initiator and the responder, they can simply check that the Diffie-Hellman 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 security association 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 security associations that do not permit such "ping" packets.

A third reason for Phase II is general security association 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 Diffie-Hellman 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 re-validate the certificate chain. A previously verified certificate chain is considered valid for the shortest of its CRL re-validate time, certificate expiration time, OCSP result validity time, *etc.* 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. TOWARDS A PROOF OF SECURITY

This section very briefly overviews our security analysis of the JFK protocol. Full details are deferred to the full analysis paper.

There are currently two main approaches to analyzing security of protocols. One is the formal-methods approach, where the cryptographic components of a protocol are modeled by "ideal boxes" and automatic theorem-verification tools are used to verify the validity of the high-level design (assuming ideal cryptography). The other is the cryptographic approach, which accounts for the fact that cryptographic components are imperfect and may potentially interact badly with each other. Here, security of protocols is proven based on some underlying computational intractability assumptions (such as the hardness of factoring large numbers, computing discrete logarithms modulo a large prime, or inverting a cryptographic hash function). The formal-methods approach, being automated, has the advantage that it is less susceptible to human errors and oversights in analysis. On the other hand, the cryptographic approach provides better soundness, since it considers the overall security of the protocol, and in particular accounts for the imperfections of the cryptographic components.

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

We separate the analysis of the "core security" of the protocol (which is rather tricky) from the analysis of added security features such as DoS protection and identity protection (which is much more straightforward). The rest of this section concentrates on the "core security" of the protocol. DoS and identity protection were discussed in previous sections.

4.1 Core security

We use the modeling and treatment of [7], which in turn is based on [6]; see there for more references and comparisons with other analytical work. Very roughly, the "core security" of a key exchange protocol boils down to two requirements:

1. 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 $K_A = K_B$.
2. No attacker can distinguish between the key exchanged in a session between two unbroken parties and a truly random value. This holds even if the attacker has total control over the communication, can invoke multiple sessions, and is told the keys generated in all other sessions.

We stress that this is only a rough sketch of the requirement. For full details see [7, 8]. We show that both JFKi and JFKr satisfy the above requirement. When these protocols are run with perfect forward secrecy, the security is based on a standard intractability assumption of the DH problem, plus the security of the signature scheme and the security of MAC as a pseudo-random function. When a party reuses its DH value, the security is based on a stronger intractability assumption involving both DH and the HMAC pseudo-random function.

We first analyze the protocols in the restricted case where the parties do not reuse the private DH exponents for multiple sessions; this is the bulk of the work. Here, the techniques for demonstrating the security of the two protocols are quite different.

4.1.1 JFKi:

The basic cryptographic core of this protocol is the same as the ISO 9798-3 protocol, which was analyzed and proven secure in [7]. This protocol can be briefly summarized as follows:

$$A \rightarrow B \quad : \quad A, N_A, g^a \tag{1}$$

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

$$A \rightarrow B \quad : \quad S_A[N_A, N_B, g^a, g^b, B] \tag{3}$$

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.

Finally, we note that JFKi enjoys the following additional property. Whenever a party P completes a JFKi exchange with peer Q, it is guaranteed that Q has initiated an exchange with P and is aware of P's existence. This property is not essential in the context of IPsec (indeed, JFKr does not enjoy this property). Nonetheless, it may be of use in other contexts.

4.1.2 JFKr:

The basic cryptographic core of this protocol follows the design of the SIGMA protocol [28] (which also serves as the basis to the signature mode of IKE). SIGMA was analyzed and proven secure in [8]. This basic protocol can be briefly summarized as follows:

$$A \rightarrow B \quad : \quad N_A, g^a \tag{1}$$

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

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

Here, neither party signs the identity of its peer. Instead, each party includes a MAC, keyed with a key derived from g^{ab}, and applied to its own identity (concatenated with N_A and N_B). JFKr enjoys the same basic core security as this protocol. In addition, JFKr adds a preliminary cookie mechanism for DoS protection (which results in adding one flow to the protocol and having the *Responder* in JFKr play the role of A), and encrypts the last two messages in order to provide identity protection. The identity protection against passive adversaries covers both parties, since the identities are sent only in the last two messages.

The next step in the analysis is to generalize to the case where the private DH exponents are reused across sessions. This is done by making stronger (but still reasonable) computational intractability assumptions involving both the DH problem and the HMAC pseudo-random function. We defer details to the full analysis paper.

5. RELATED WORK

The basis for most key agreement protocols based on public-key signatures has been the Station to Station (StS)[11] protocol. In its simplest form, shown in Figure 1, this consists of a Diffie-Hellman 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 3-message protocol, shown in Figure 2. Other forms of authentication may be used instead of public key signatures (*e*.g., Kerberos[37] 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[13] in terms of messages and computation, it suffers from some obvious DoS vulnerabilities.

5.1 Internet Key Exchange (IKE)

The Internet Key Exchange protocol (IKE)[15] is the current IETF standard for key establishment and SA parameter negotiation.

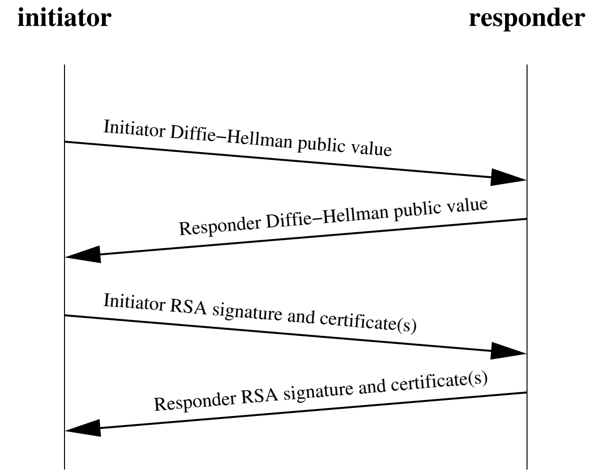

Figure 1: 4-message Station to Station key agreement protocol.

IKE is based on the ISAKMP [33] 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 [27].

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 Diffie-Hellman 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 Diffie-Hellman 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[37] and handheld authenticators. It should also be noted that IKE can support other key establishment mechanisms (besides Diffie-Hellman), 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] denial of service (DoS) attacks against both computational and memory resources[42]. 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 mes-

[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 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 Diffie-Hellman secret. This method of peer authentication is based on the Sign-and-Mac design [28].

[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.

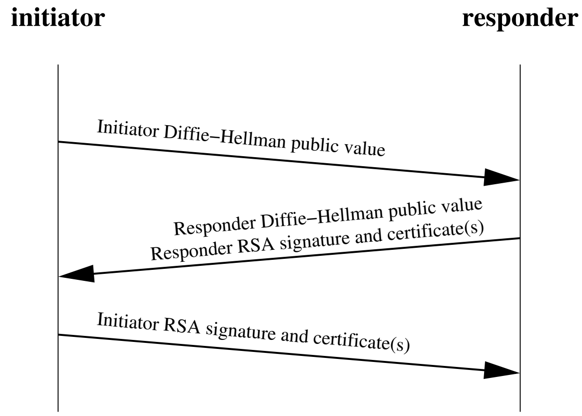

Figure 2: 3-message Station to Station key agreement protocol.

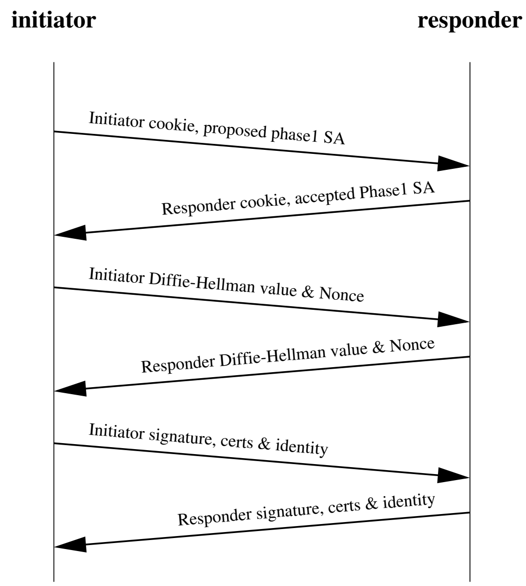

Figure 3: IKE Main Mode exchange with certificates.

sage 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 employ mechanisms similar to those employed in countering TCP SYN attacks[17, 9, 40]. 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 Diffie-Hellman 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, *etc.* 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 Diffie-Hellman 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 re-keying. 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 & 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, *etc.*). 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, *etc.*), 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 5 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 *vs.* Main mode, Phase I / II distinction), both the protocol specification and the implementations tend to be bulky and fairly complicated. These are undesirable properties for a critical component of the IPsec architecture.

Several works (including [12, 26, 25]) 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, *etc.*), 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

[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.

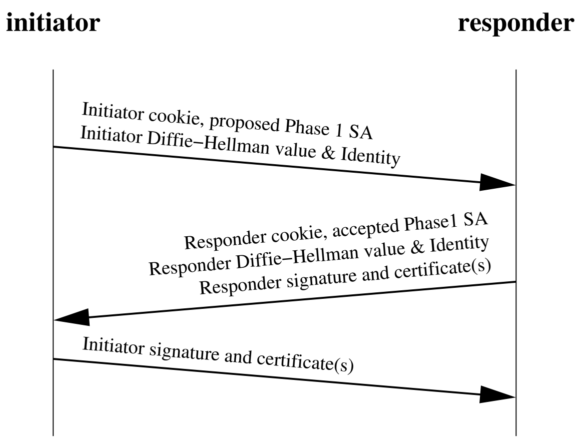

Figure 4: IKE Aggressive Mode exchange with certificates.

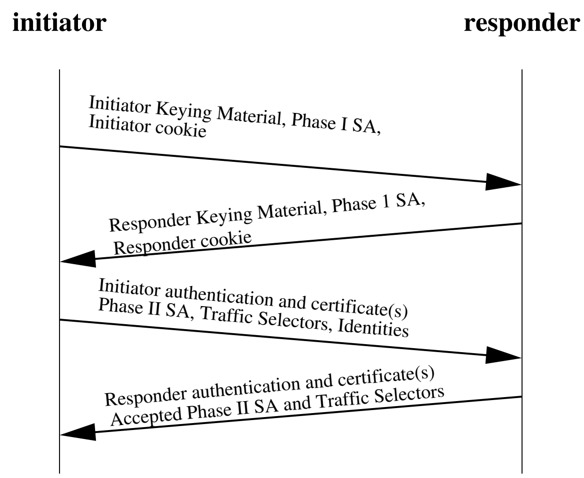

Figure 5: IKEv2 protocol exchange.

to what is common practice when using SSL/TLS[10] 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 [34, 36]. No less than 13 different sub-protocols 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[31, 32] and, more generally, in public key authentication protocols[30, 21]. Various recommendations on protocol design include use of client puzzles[23, 3], stateless cookies[39], forcing clients to store server state, rearranging the order of computations in a protocol[18], and the use of a formal method framework for analyzing the properties of protocols with respect to DoS attacks[35]. The advantages of being stateless, at least in the beginning of a protocol run, were recognized in the security protocol context in [22] and [2]. The latter presented a 3-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[16] 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 4 messages shown; however, if the responder detects a DoS attack, it can start requiring the extra roundtrip. One claimed benefit of this extra roundtrip is the ability to avoid memory-based DoS 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 roundtrip 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 roundtrip, 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[10].
- IKEv2 supports legacy authentication mechanisms (in particular, pre-shared 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) which 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[19], 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 security association 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[24], first introduced the concept of cookies to counter "blind" denial of service attacks. The protocol itself is a 6-message variation of the Station to Station 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 re-keying, a two-message exchange can be used to request a uni-directional SPI (thus, to completely re-key, 4 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 4 messages and provided DoS protection via stateless cookies.

SKEME[27] 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[5] was an early proposal for an IPsec key management mechanism. It uses long-term Diffie-Hellman 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, *i.e.,* 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 re-keying, the base version lacks any provision for PFS. It was later provided via an extension [4]. 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 Diffie-Hellman 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 Diffie-Hellman certificate). The latter can be used to mount a DoS attack on a receiver, by forcing them to retrieve and verify a Diffie-Hellman certificate, and then compute the Diffie-Hellman shared secret.

The Host Identity Payload (HIP)[38] 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 Diffie-Hellman value over a period of time, to handle a high rate of sessions. For re-keying, 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 denial of service 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, *etc.* We show that DoS-resistance and perfect forward secrecy 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.

7. ACKNOWLEDGEMENTS

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.

8. REFERENCES

[1] A. Arsenault and S. Farrell. Securely available credentials - requirements. Request for Comments 3157, Internet Engineering Task Force, Aug. 2001.

[2] T. Aura and P. Nikander. Stateless connections. In *Proc. of International Conferenec on Information and Communications Security (ICICS '97), Lecture Notes in Computer Science volume 1334*, pages 87–97. Springer, November 1997.

[3] T. Aura, P. Nikander, and J. Leiwo. DOS-resistant authentication with client puzzles. In *Proc. of the 8th International Workshop on Security Protocols*, April 2000.

[4] A. Aziz. SKIP extension for perfect forward secrecy (PFS). Internet Draft, Internet Engineering Task Force, August 1996.

[5] A. Aziz and M. Patterson. Simple Key Management for Internet Protocols (SKIP). In *Proc. of the 1995 INET conference*, 1995.

[6] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Proc. of the Crypto conference*, August 1993.

[7] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Proc. of the Eurocrypt conference*, May 2001.

[8] R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-based Key-Exchange Protocol. In *Proc. of the Crypto conference*, August 2002.

[9] CERT. Advisory CA-96.21: TCP SYN Flooding. ftp://info.cert.org/pub/cert_advisories/CA-96.21.tcp_syn_flooding, September 1996.

[10] T. Dierks and C. Allen. The TLS protocol version 1.0. Request for Comments (Proposed Standard) 2246, Internet Engineering Task Force, Jan. 1999.

[11] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[12] N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPSec. `http://www.counterpane.com/ipsec.html`, 1999.

[13] L. Gong. Efficient Network Authentication Protocols: Lower Bounds and Optimal Implementations. *Distributed Computing*, 9(3):131–145, 1995.

[14] D. Gustafson, M. Just, and M. Nystrom. Securely available credentials - credential server framework. Internet Draft, Internet Engineering Task Force, Aug. 2001. Work in progress.

[15] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). Request for Comments (Proposed Standard) 2409, Internet Engineering Task Force, Nov. 1998.

[16] D. Harkins, C. Kaufman, S. Kent, T. Kivinen, and R. Perlman. Proposal for the IKEv2 Protocol. Internet Draft, Internet Engineering Task Force, April 2002. Work in progress.

[17] L. Heberlein and M. Bishop. Attack Class: Address Spoofing. In *Proceedings of the 19th National Information Systems Security Conference*, pages 371–377, October 1996.

[18] S. Hirose and K. Matsuura. Enhancing the resistance of a provably secure key agreement protocol to a denial-of-service attack. In *Proc. of the 2nd International Conference on Information and Communication Security (ICICS '99)*, pages 169–182, November 1999.

[19] P. Hoffman. Features of Proposed Successors to IKE. Internet Draft, Internet Engineering Task Force, April 2002. Work in progress.

[20] IEEE. Entity authentication mechanisms — part 3: Entity authentication using asymmetric techniques. Technical Report ISO/IEC IS 9798-3, ISO/IEC, 1993.

[21] M. Jakobsson and A. Juels. Proofs of work and bread pudding protocols. In *Proc. of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security*, September 1999.

[22] P. Janson, G. Tsudik, and M. Yung. Scalability and flexibility in authentication services: the KryptoKnight approach. In *Proc. of IEEE INFOCOM*, pages 725–736, April 1997.

[23] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proc. of the Network and Distributed Systems Security Symposium (NDSS '99)*, pages 151–165, February 1999.

[24] P. Karn and W. Simpson. Photuris: Session-key management protocol. Request for Comments 2522, Internet Engineering Task Force, Mar. 1999.

[25] C. Kaufman et al. Code-preserving Simplifications and Improvements to IKE. Internet Draft, Internet Engineering Task Force, July 2001. Work in progress.

[26] C. Kaufman and R. Perlman. Analysis of IKE. In *IEEE Transactions on Network Computing,*, November 2000.

[27] H. Krawczyk. SKEME: A Versatile Secure Key Exchange Mechanism for Internet. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, February 1996.

[28] H. Krawczyk. The IKE-SIGMA Protocol. http://www.ee.technion.ac.il/˜hugo/sigma.html, November 2001.

[29] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: keyed-hashing for message authentication. Request for Comments 2104, Internet Engineering Task Force, Feb. 1997.

[30] J. Leiwo, P. Nikander, and T. Aura. Towards network denial of service resistant protocols. In *Proc. of the 15th International Information Security Conference (IFIP/SEC)*, August 2000.

[31] K. Matsuura and H. Imai. Resolution of ISAKMP/Oakley key-agreement protocol resistant against denial-of-service attack. In *Proc. of Internet Workshop (IWS '99)*, pages 17–24, February 1999.

[32] K. Matsuura and H. Imai. Modified aggressive mode of Internet key exchange resistant against denial-of-service attacks. *IEICE Transactions on Information and Systems*, E83-D(5):972–979, May 2000.

[33] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet security association and key management protocol (ISAKMP). Request for Comments (Proposed Standard) 2408, Internet Engineering Task Force, Nov. 1998.

[34] C. Meadows. Analysis of the Internet Key Exchange protocol using the NRL protocol analyzer. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 216–231, May 1999.

[35] C. Meadows. A formal framework and evaluation method for network denial of service. In *Proc. of the 12th IEEE Computer Security Foundations Workshop*, pages 4–13, June 1999.

[36] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proc. of DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, pages 237–250. IEEE Computer Society Press, January 2000.

[37] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos Authentication and Authorization System. Technical report, MIT, December 1987.

[38] R. Moskowitz. The Host Identity Payload. Internet Draft, Internet Engineering Task Force, July 2001. Work in progress.

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

[40] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on tcp. In *IEEE Security and Privacy Conference*, pages 208–223, May 1997.

[41] Y. Sheffer, H. Krawczyk, and B. Aboba. PIC, a pre-IKE credential provisioning protocol. Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

[42] W. A. Simpson. IKE/ISAKMP Considered Harmful. *USENIX ;login:*, December 1999.