Internet Draft   Editor: Paul Hoffman
draft-ietf-ipsec-soi-features-01.txt   VPN Consortium
May 31, 2002
Expires in six months

Features of Proposed Successors to IKE

Abstract

This document describes many features of the proposals for the successor to IKEv1. The purpose of the document is to help the IPsec Working Group decide which features they want for the next protocol. The document focuses on how those features are instantiated in two proposals, IKEv2 and JFK, but other ideas for features and options are also included. Most of the material in this document comes from many of the authors of the JFK and IKEv2 proposals.

This document is meant to help the Working Group choose which features it finds important for the successor to IKE. It should be short-lived and is not expected to become an RFC.

1. Introduction

The IPsec Working Group is actively considering the successor to IKE (also known as "son of IKE"). The Working Group is evaluating the requirements for keying IPsec, as well as looking at the positive and negative lessons learned from many years of IKE development and deployment.

Many of the categories of features described in this document overlap, so it is not always possible to cleanly break out the features into stand-alone units.

In this document, the terms "Alice" and "the initiator" are used interchangeably, as are "Bob" and "the responder".

A summary of the features covered is listed near the end of this document.

1.1 Selecting features for the successor to IKE

At this point, two proposals for the successor to IKE have been discussed most heavily: IKEv2 and JFK. Each proposal has a myriad of features. In many cases, the two proposals have chosen to address a particular feature the same way; for many other features, the two proposals differ.

It is important to note that this document describes the features of the two protocols today, as well as features and options that are not described in either protocol. Most of the features listed here could be added (or removed) from either protocol. Thus, if a particular feature is important to you and it exists in one protocol but not the other, you should not assume that the other protocol cannot include the feature.

It is also important to note that there are often more than two choices for a particular feature. For example, there are many different ways that identity protection could be done, not just the two ways described in the current proposals. The Working Group is free to consider all options, not just those embodied in the current proposals.

This document attempts to list as many sensible options as possible for the features listed. Some sensible options may have been inadvertently omitted.

For some features, the description is just an exposition about how the two protocols differ. This does not mean that "JFK will always do this" or "IKEv2 will always do that". Instead, it is just a shorthand way of describing two views of the feature. In most cases, either protocol could be changed to work differently.

When this document talks about how either of the two proposals deals with a feature, that discussion is based on the latest drafts of the two proposals: draft-ietf-ipsec-ikev2-02.txt and draft-ietf-ipsec-jfk-03.txt. JFK has two variants: JFKr and JFKi. When the features in them differ, this document discusses the particular variant. Note that the two proposals have made significant changes in they way they approach some of the features.

1.2 Choosing a starting point

As discussed above, most of the features in this document can be implemented in different ways. Once the IPsec Working Group has picked the features it wants, it then must decide how to move forward and create a protocol. The three likely choices are to start with IKEv2, start with JFK, or start from scratch.

The authors of JFK and IKEv2 have stated that they are quite open to adding or changing major parts of their protocols. Some features are so much part of the core of a protocol that the cannot be changed, however. The features that are core differences between the two protocols are:

- Whether the protocol has one phase or two phases

- How DoS attacks are thwarted

- The use of shared secret authentication

Thus, the Working Group may have to select the protocol to work from based on these features, and can mix and match the other features from that starting point.

2. Cryptographic features

2.1 Identity protection: who is protected, kinds of attacks

There are two types of attacks that can mounted to find out the identity of one or both of the parties in a key exchange protocol:

- Passive attacks look for identities that passed in the clear during the negotiation. In a passive attack, neither party can detect that the identity is being seen.

- Active attacks find identities by being a man-in-the-middle or by replacing the initiator or responder in the negotiation. The attacker proceeds through the key negotiation with the attackee until the attackee has revealed its identity. In a well-designed system, the negotiation will fail after the attackee has revealed its identity because the attacker cannot spoof the identity of the originally-intended system. The attackee might then suspect that there was an attack because the other side failed before it gave its identity. Therefore, an active attack cannot be persistent because it would prevent all legitimate access to the desired IPsec system.

In key negotiations, the IP address of each IPsec system is always known to a passive or active attacker. The identities being protected by the key negotiation system are those in the certificates or identification payloads.

Typically, an IPsec system that is at the same IP address over a long period of time does not need identity protection because an attacker can use other means (such as traceroute and reverse DNS lookup) to determine its identity. Identity protection is most useful to users with dynamic

IP addresses, usually users whose IP address is assigned by DHCP or by a variety of different ISPs.

The four cases for a negotiation are:

Stationary initiator, stationary responder: This is typical of gateway-to-gateway VPNs. Identity protection would not achieve much here.

Mobile initiator, stationary responder: This is a typical remote-access scenario. Even though the responder's identity can probably be determined by other means, the initiator can get value from identity protection.

Stationary initiator, mobile responder: For the initiator to be able to find the responder, there must have been some recent prior contact between the two parties (or other out-of-band mechanisms for location). This could, for example, be a rekeying of an existing SA. In this case, the responder would want its identity protected.

Mobile initiator, mobile responder: For the initiator to be able to find the responder, there must have been some recent prior contact between the two parties (or other out-of-band mechanisms for location). This could, for example, be a rekeying of an existing SA. In this case, each side would want its identity protected.

2.1.1 Proposals for protection

In IKEv2 and JFKr, there is no passive attack possible, and an active attack is possible against the initiator's identity. Such an attack will reveal the identity but will cause the negotiation to fail in the next message.

In JFKi, there is a passive attack on the responder's identity, but no active attack possible on the initiator.

It has also been proposed that the protocol allow no passive attack, and allow an active attack on the responder's identity. This can be done in many ways, such as extending message 2 to have encrypted and unencrypted parts (making the entire protocol three messages), by having message
3 come from the responder instead of from the initiator, or by making
the protocol five messages long, with message 3 being a place-holder.

2.2 Perfect forward secrecy (PFS)

"Perfect forward secrecy" is the property that in the event of a compromise of the system, past conversations recorded by the attacker cannot be decrypted.

JFK incorporates "imperfect forward secrecy" into its base protocol. That is, the essential component (a Diffie-Hellman exchange) is not optional. However, as part of its DoS-resistance capability, the server may reuse its exponential as often as it wants. The degree of perfection of the forward secrecy, then, is dependent on the frequency of generation of new exponentials, which in turn is dependent on details of the implementation, the intensity of any DoS attack, and policy as set by the site administrator.

IKEv2 uses a mandatory Diffie-Hellman exchange in Phase 1. Assuming that both parties choose new Diffie-Hellman exponentials each time they do Phase 1, there will be forward secrecy between IKE SAs. In addition, IKEv2 allows perfect forward secrecy for Phase 2 SAs by allowing an optional Diffie-Hellman exchange during rekeying of IPsec SAs in Phase 2**. Performance can be traded off against PFS in IKEv2 as well. The IKEv2** specification describes how to reuse Diffie-Hellman exponents across Phase 1 sessions, and change them periodically for imperfect forward secrecy.

A protocol could always insist on doing PFS every time.

2.3 Authentication styles

IKEv2 supports two forms of authentication: certificates and shared secret. JFK supports one form of authentications: certificates.

Any protocol that supports certificates also supports identification of individuals without certificate hierarchies through the use of self-signed certificates. If the receiving system loads a self-signed certificate based on out-of-band communications, the party that has the private key associated with the public key in the certificate can authenticate to the receiving system.

IKEv2 indirectly supports non-certificate, non-shared-secret authentication (commonly called "legacy authentication") though the use of the ID_KEY_ID identity type and the shared secret authentication. The authentication system must create two temporary tokens, one of which is used as the identity and the other is used as the secret. The IKEv2 system must have some way of securely querying the authentication system with the identity token and receiving back the secret token.

2.4 Number of crypto operations

2.4.1 JFKr

In JFKr, the following crypto operations occur:

Message 1:
- Choose a Diffie-Hellman exponential

Message 2:
- Choose a Diffie-Hellman exponential
- Create an HMAC

Message 3:

- Compute Diffie-Hellman shared secret
- Sign
- Encrypt
- Create an HMAC

Message 4:
- Compute Diffie-Hellman shared secret
- Validate the first HMAC
- Decrypt
- Validate signature
- Sign
- Encrypt
- Create an HMAC
- Validate the second HMAC

The initiator then has to:
- Decrypt
- Validate a signature
- Validate an HMAC

The initiator and responder can each choose to reuse a Diffie-Hellman exponential if they want to extend the forward secrecy window.

2.4.2 JFKi

In JFKi, the following crypto operations occur:

Message 1:
- Choose a Diffie-Hellman exponential

Message 2:
- Choose a Diffie-Hellman exponential
- Sign
- Create an HMAC

Message 3:
- Compute Diffie-Hellman shared secret
- Validate a signature
- Sign
- Encrypt
- Create an HMAC

Message 4:
- Compute Diffie-Hellman shared secret
- Validate HMAC
- Decrypt
- Validate signature
- Encrypt

The initiator then has to:
- Decrypt

- Validate a signature

The initiator and responder can each choose not to choose a Diffie-Hellman exponential if they want to extend the forward secrecy window.

2.4.3 IKEv2

In Phase 1 of IKEv2, the following crypto operations occur:

Phase 1, message 1:
- Choose a Diffie-Hellman exponential

Phase 1, message 2:
- Choose a Diffie-Hellman exponential

Phase 1, message 3:
- Compute Diffie-Hellman shared secret
- Sign (for certificates) or HMAC (for shared secret)
- Encrypt
- Create an HMAC

Phase 1, message 4:
- Compute Diffie-Hellman shared secret
- Validate an HMAC
- Decrypt
- Validate signature (for certificates) or HMAC(for shared secret)
- Sign (for certificates) or HMAC (for shared secret)
- Encrypt
- Create an HMAC

The initiator then has to:
- Validate an HMAC
- Decrypt
- Validate signature (for certificates) or HMAC(for shared secret)

In Phase 2 of IKEv2, when creating a new IPsec SA, the following crypto operations occur:

Phase 2, message 1:
- Optionally choose a Diffie-Hellman exponential
- Encrypt
- Create an HMAC

Phase 2, message 1:
- Decrypt
- Optionally choose a Diffie-Hellman exponential
- Encrypt
- Create an HMAC

The initiator then has to:
- Validate an HMAC

- Decrypt

In IKEv2 Phase 2, the Diffie-Hellman exponentials must be given if they were negotiated in Phase 1.

2.4.4 Summary of the number of crypto operations

In all of the proposals, each side must choose a Diffie-Hellman exponential and compute the Diffie-Hellman shared secret. In the tables below, the first number is for the initiator, the second number is for the responder.

	Sign	Validate signature	Symmetric crypt	Hash
JFKr	i:1 r:1	i:1 r:1	i:2 r:2	i:2 r:4
JFKi	i:1 r:1	i:2 r:1	i:2 r:2	i:1 r:2
IKEv2 sig	i:1 r:1	i:1 r:1	i:2 r:2	i:2 r:2
IKEv2 secret	i:0 r:0	i:0 r:0	i:2 r:2	i:3 r:3

In other words, the basic difference between JFKr and IKEv2 in signature mode is that JFKr requires two more hashes for the responder.

For rekeying an existing IPsec SA, the summary looks like:

	Sign	Validate signature	Symmetric crypt	Hash
JFKr	i:1 r:1	i:1 r:1	i:2 r:2	i:2 r:4
JFKi	i:1 r:1	i:2 r:1	i:2 r:2	i:1 r:2
IKEv2	i:0 r:0	i:0 r:0	i:2 r:2	i:2 r:1

In other words, for rekeying, JFKr requires two more signatures, two more signature validations, and three more hashes.

2.5 Plausible deniability

If Alice and Bob are using a public key cryptosystem that supports digital signatures (such as RSA or DSA) as part of their authentication and key establishment protocol, it may be possible for Alice to prove to a third party, Carol, that Bob did indeed participate in a protocol exchange with her. If Alice can do so, the exchange is said to be "non-repudiable" (that is, Bob cannot repudiate having done it).

On the other hand, it is possible to construct the protocol in such a way that, while both parties are convinced at the end of the protocol exchange that the other is present and has authenticated, they cannot prove this to a third party; that is, Carol cannot determine whether Bob really did participate in the exchange or Alice constructed the evidence. Bob is then said to have "plausible deniability" in having participated in the exchange with Alice, as far as anyone else can discern.

Both IKEv2 and JFKr provide plausible deniability to both Alice and Bob, since in neither protocol is the identity of the peer signed by either party. Notice that shared-secret authentication schemes are by nature repudiable because either party can impersonate the other).

JFKi does not provide plausible deniability to either Alice or Bob, because each must sign with their private key with a quantity which includes the other's name.

2.6 Formal proofs of security

JFK has an extensive cryptographic proof section in the specification. IKEv2 doesn't contain any cryptographic proof. The signature mode of IKEv2 is similar to the signature mode of IKEv1; IKEv1 was analyzed and proven secure in [CK02].

3. Protocol mechanics

3.1 DoS protection

Each protocol uses an anti-clogging token as a DoS protection mechanism. The responder (if under attack in IKEv2, and always in JFK) computes the token using something known only to him and other things the initiator provided in the request. This binds Alice to the exchange and if the initiator can produce the token again in a subsequent message along with the information that is bound in the token then the responder has a strong reason to believe the initiator is a real, live, peer and not a scripted attack.

Two different types of DoS attacks are thwarted by JFK and IKEv2: attacks that would cause the protocol itself to exhaust memory or CPU, and attacks that could cause the lower-level packet handler in the implementation to exhaust memory by sending incomplete UDP packet fragments.

For basic DoS attacks that would cause the protocol itself to exhaust memory or CPU, the difference in approach between the two protocols is that JFK tries to be completely stateless after sending message 1 and is always a 4 message exchange, while IKEv2 can be "stateless" in six messages or "stateful" in four messages. In IKEv2, the decision of when to switch from stateful to stateless is at the discretion of the implementation when it feels it has too many outstanding, larval exchanges, or when it feels it is "under attack".

In the UDP fragmentation attacks, the lower-level UPD fragment handler must have some way of being smart when its fragment buffer gets nearly full. Note that in both protocols, message 3 can be legitimately fragmented. JFK specifies a method where the lower-level system checks for a value in the ip_id field of the UDP packets. In IKEv2, the IKEv2 system informs the lower-level system to only accept UDP fragments from IP addresses from which it has received a valid cookie, and to refuse to accept UDP fragments from all other IP addresses.

3.1.2 Commentary on IKEv2 stateless cookies

[PK00] explains how to do stateless cookies without adding additional messages, by repeating information from message 1 in message 3. It is possible to do stateless cookies without adding an additional two messages. However, IKEv2 says instead that if Bob feels he is under attack and would like to use a stateless cookie mechanism, he rejects message 1 and sends a stateless cookie, and Alice repeats message 1, this time with the stateless cookie.

The reasons for doing this rather than the mechanism proposed in [PK00], adopted by JFK, which avoids the extra 2 messages are:

- This makes message 3 shorter. If Bob does not need the stateless cookie mechanism, there will be fewer bits transmitted

- It is complex and implementation-costly to have message 3 be partly encrypted and partly unencrypted, as would be necessary to do the [PK00] mechanism.

- Keeping message 1 (and the repeated message 1 plus the cookie) small enough not to require fragmentation avoids a fragmentation DOS attack. Bob needs to keep state to reassemble fragmented packets. Using the stateless cookie mechanism with small messages until a cookie is returned allows an implementation to defend against fragmentation DoS attack. For this to work, IKEv2 gives hints to the reassembly code about which IP addresses should be preferred for state. Since IKEv2 will not require fragmentation until the cookie is returned and Alice's IP address can be added to the preferred category, the limited reassembly buffers can be reserved for reassembling packets from IP addresses which have returned valid cookies.

3.1.2 Commentary on JFK's reuse of exponentials

JFK's DoS-resistance is, more generally, a mechanism for limiting the CPU consumption of the participants at the time of SA creation. An implementation can have a low-priority background process that generates a queue of exponentials. When creating an SA, the process can use an exponential from the queue. If the queue is empty, either a new exponential is generated or the system reuses an existing exponential, based on a limit set by the administrator. This avoids the necessity to make an explicit determination that there is an attack in progress; additionally, it is well-suited to situations where there may be a sudden surge of new SAs, such as when a security gateway reboots.

3.2 Number of messages in all scenarios

As described in section 2.1, the anti-DoS mechanism in JFK requires no additional messages, while the anti-DoS mechanism in IKEv2 requires an additional round-trip when the responder believes it is under attack. Thus, the number of messages for a normal JFK setup is four, and the number for a normal IKEv2 setup is four or six. However, these numbers assume that the cryptographic suites are agreed to by both parties on the first try.

IKEv2 and JFK both use 4 messages as the base operation. In messages 1 and 2 there is a Diffie-Hellman exchange. In messages 3 and 4 identities are exchanged, along with proof of knowledge of the private key, encrypted and integrity protected with a function of the Diffie-Hellman key.

In both IKEv2 and JFK, Alice chooses a Diffie-Hellman group in message 1**. In IKEv2, if she chooses a group that Bob does not support, the** protocol fails and Alice must start again, causing the use of an extra two messages. In JFK, if she chooses a group that Bob does not support, but can use the one Bob suggested (in g^r), she can proceed directly to message 3 by using the same Ni and a new g^i; otherwise, the protocol fails and Alice must start again, causing the use of an extra two messages.

3.3 Size of messages

In both protocols, each message is fairly small unless it contains a certificate. The differences in the size of the messages is small relative to items that are inherently long, such as Diffie-Hellman exponentials.

3.4 Preferred ID for responder

In JFK and IKEv2, the initiator can include a payload is an indication to the responder as to what identity (and corresponding key material) the responder should use to authenticate to the initiator. The responder may ignore this hint. In JFKr and IKEv2, this value is encrypted in message 3; in JFKi, it is sent in the clear in message 1, thereby allowing a passive attack on the responder's likely identity.

3.5 Birth certificates

Assume Alice and Bob have an SA established, and that Bob crashes and restarts. Alice will send a packet into an SA that Bob doesn't know about. Birth certificates are a method for letting Alice know that the SA is no longer alive.

It is desirable that such a mechanism be inexpensive for Bob. If a mechanism required a lot of computation, for instance, having Bob sign a customized message saying, "This SA didn't decrypt properly. Perhaps it's because it's a reused SA number from before I crashed" or "unknown SPI", then there is an easy DOS attack on Bob's limited computation by sending him bogus messages. If Bob sends totally unauthenticated messages, it is difficult for Alice to act on them, since an active attacker could send her "no such SPI" messages to cause her to close connections.

The idea of a birth certificate, originated in the IPsec Working Group by Bill Sommerfeld, is that Bob should have a monotonically increasing number that increases each time Bob restarts. When Bob restarts, he signs a message saying, "this is my incarnation number". When Bob creates an SA, Bob optionally sends his birth certificate. Alice can optionally keep this certificate with her SA. Then, if Alice ever receives an error message from Bob with a birth certificate with a higher incarnation number, she can remove all SAs associated with Bob's previous incarnation.

This mechanism is somewhat overlapping with the "initial-contact" mechanism in IKEv2. Discussion in the WG suggested that birth certificates could replace initial-contact, but some people said that both were valuable.

Neither IKEv2 nor JFK have this feature.

4. One or two phases

4.1 Control channel vs. separate protocols

In IKEv2, Phase 1 is used as a control channel for IPsec SAs. After the Phase 1 has been set up, it can be used to:

- create new IPsec SAs

- delete existing IPsec SAs

- rekeying existing IPsec SAs

- send protected notifications

- perform dead peer detection

In JFK, there is no Phase 2, and separate protocols are specified for the control functions. The JFK proposal describes the following:

- using JFK again to create new IPsec SAs

- deleting existing SAs by rekeying an existing SA with AH_BYPASS or ESP_BYPASS

- there is no need to rekey existing SAs because new ones are created

- no protected notifications are specified in JFK

- dead peer detection is done by noting the time of the last packet received

Proposals have also been made for additional SA control functions such as keep-alives.

4.2 Creating multiple SAs for a single pair of entities

In JFK, each SA is created and managed independently. A pair of entities can have multiple SAs by running the JFK protocol multiple times, each time picking a new SA identifier.

In IKEv2, a single IKE SA can be used to create multiple IPsec SAs for the same pair of entities if the IPsec SAs use the same credentials; if different credentials are needed for some IPsec SAs, new IKE SAs must be created. In addition, a pair of entities can have multiple SAs by creating new IKE SAs (that is, by creating a new Phase 1).

4.3 Dead peer detection

JFK does not incorporate any intrinsic facilities for dead peer detection. Instead, the draft suggests an outboard approach. A dead peer is defined as one from which no packets have been received successfully for some site-dependent length of time. An IPsec node may attempt to elicit a packet by sending one of its own, such as an ICMP "Echo Request" message. To enable this, it is entitled to request an SA that permits such messages, and to decline any SA that does not include such permission.

IKEv2 relies on a similar strategy. However, instead of relying on ordinary SA traffic, it uses responses (or the lack thereof) from authenticated Phase 1 traffic. IKEv2 explicitly allows a peer to decide that other peer has died because of lack of responses to lack of acknowledgment to a Phase 1 message, even an empty informational exchange.

4.4 SA deletion

In JFK, a new IPsec SA that specifies an SPD identical to the SPD of an existing SA overwrites the old one. To delete an SA, create a new one with the ESP_BYPASS or AH_BYPASS ciphersuites.

In IKEv2, both IPsec SAs and IKE SAs are deleted with a delete payload.

4.5 SA rekeying

In IKEv2 and JFK, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. In both protocols, SAs are rekeyed by creating a new equivalent SA, then deleting the old SA. In IKEv2, this applies to both IKE SAs and the IPsec SAs.

4.6 Authenticated error messages

In some situations, either of the protocol participants may detect an error that can be recovered from if the peer modifies its behavior. (Errors that are non-recoverable include implementation failures, such as a mis-formatted message and so on.) Even for non-recoverable errors, it is sometimes desirable to send an error indication to the peer, so the recipient can take some kind of corrective action (or at least not keep doing whatever is causing the error). In those situations, the protocol may support the transmission of error messages. The question is whether these messages need to be cryptographically protected.

There are three types of attacks on error messages:

a) Information leakage: if the error message includes information identifying one of the protocol participants, and identity protection is desired, the error message should obviously be encrypted. Likewise, receipt of an error message should not cause the receiving party to perform an action that discloses either party's identity.

b) Downgrade attacks: if the error message indicates a choice in encryption algorithms (or otherwise protocol "strength") and the message is not authenticated, it may be possible for an active attacker to cause a weaker cipher (or other protocol parameter) to be used, making it easier to mount a cryptographic attack. This particular attack is possible against version 2 of the SSL protocol.

c) Denial of service attacks: error messages that cause an otherwise valid exchange to be dropped (perhaps with catastrophic consequences, such as a black-hole SPD entry to be installed).
An attacker that sits in the
middle of the network can always deny service simply by dropping packets. Of more interest are situations where an attacker that is not on the direct communication path (and thus cannot directly perform a protocol DoS attack) can nonetheless cause protocol failure by sending fake error messages, perhaps after some minimal guessing or precomputation.

There are, generally, two kinds of error messages:

- Error messages exchanged in the course of the key exchange protocol itself. If these are "fatal" for the protocol, the messages should either contain enough randomness (such as in the form of cookies) to avoid blind attacks, or the messages should be authenticated if they occur late enough in the protocol exchange that key material has been established. If they are not fatal (such as if they indicate a set of acceptable options), the same options as above are available; alternatively, it is acceptable that these messages are not strongly protected if their contents are included in some integrity verification later on (such as the JFKr authentication of GRPINFOr which is sent in message 2, but is included in the signature of message 3 in the protocol).

- Error messages having to do with events after the protocol exchange has been completed (such as an "SA delete" message). Because the authentication has been completed, it is fairly straightforward to provision for some kind of protected control channel between the two peers, using key material and other parameters already negotiated. Whether this control channel is provided as part of the base protocol, or it's implemented as a separate protocol is a protocol design decision.

In IKEv2, most error messages and notifications are protected under an IKE SA, and their delivery is confirmed. There are two error messages that are not (and cannot be) protected in this way: errors in response to the message 1 in the exchange (requesting a cookie, or indicating an unacceptable SA payload), and notifications sent when a host receives an ESP or AH packet with an unknown SPI (possibly because of a reboot). In the former case, an attacker would need to know (or predict) the Initiator cookie, before he can fool the Initiator into accepting an error message. As an additional precaution, the Initiator may wait for a short period of time (e.g., the retransmission timer) before dropping the exchange (or modifying his behavior), in case another packet (not an error message) arrives. This situation may arise when an attacker can observe traffic between two parties but cannot prevent messages from going through (such as on a wireless link). In the latter case (sending a notification in response to an unknown SPI), care must be taken in how the recipient responds: simply re-negotiating a new SA pair can lead to a blind DoS attack. One solution is to examine the last time an ESP/AH packet was received from the peer that purportedly send the "unknown SPI" message, as well as invoke any keep-alive mechanism. (Notice that this last item argues for a keep-alive mechanism that is outside the key exchange protocol and is protected by ESP/AH; otherwise, the same potential for a DoS attack exists.)

The exact same considerations are present in JFK; an error notification returned as a response to message 1 cannot be protected. Errors in message 3 (errors sent from the Initiator to the Responder) can be protected, unless they indicated a failure in the key exchange itself (such as a shared key cannot be computed). Errors in message 4 are authenticated under key material already exchanged. JFK itself does not provide for any other error or notification messages. Peer keep-alives or other similar notifications must be implemented as protocols using the ESP/AH SAs already established, or their own separate mechanisms (if appropriate). In the former case, those messages are protected under the IPsec SAs established.

4.7 Authenticated informational messages

JFK has no authenticated informational messages. IKEv2 has authenticated informational messages after an IKE SA has been set up.

5. SA creation style

5.1 Cryptographic agreement

In order to set up an IPsec SA, the two parties have to agree on two separate cryptographic suites: one suite for key exchange, another for use in the IPsec SA itself. In both protocols, agreeing on a Diffie-Hellman group used for key exchange is different than agreeing on the cryptographic material used in the IPsec SA. In both protocols, the two peers want to agree on a single suite of security parameters for the key negotiation, and a single set of security parameters for IPsec; the two sets might be very different.

5.1.1 Agreement of key exchange keys

In both IKEv2 and JFK, Alice chooses a Diffie-Hellman group in message 1.

IKEv2 provides a form of negotiation for keys that allows the two parties to agree on the most preferred parameters. If both can do AES and 3DES but one is preferred (for instance if 3DES had hardware acceleration while AES did not), then they can agree on the preferred parameter. If Alice chooses a group that Bob does not support, the protocol fails and Alice must start again. In this failure case, Bob does not tell Alice what groups he accepts.

In JFK, Bob states one crypto suite that contains a single encryption algorithm, a single hash algorithm, and one or more Diffie-Hellman groups. If Alice doesn't accept Bob's suite, Alice will give up and end the communication. In message 1, if Alice chooses a group that Bob does not support, but she can use the one Bob suggested (in his g^r), she can proceed directly to message 3 by using the same Ni and a new g^i. Even if Alice can't use the group from Bob's g^r, Bob tells Alice which group he supports in his failure notification to her.

The difference in styles between IKEv2 and JFK affects changing encryption and hash algorithms used in key exchange. In IKEv2, Alice can add this algorithm to her list offered to Bob; if Bob doesn't know about it yet, he will use one of the older, less-wonderful algorithms. JFK assumes that no different algorithm will be needed for key exchange; if it is, Bob must communicate this change out-of-band to all of the parties that initiate to him. In this respect, IKEv2 is meant to allow negotiation; JFK is meant to allow simplicity.

5.1.2 Agreement of IPsec SA cryptographic algorithms

If one peer is only willing to accept a single suite in JFK or only one option of each of the parameters in IKEv2, then that peer is said to be "demanding". In JFK, Bob "proposes/demands" a single suite; in IKEv2, Alice "proposes/demands" one or more options for each part of the set of parameters.

IKEv2 SA negotiation allows the two parties to agree on the most preferred parameters, the same as it does for key negotiation.

In JFK SA negotiation, Alice states one crypto suite; if Bob doesn't accept that suite, he rejects it and Alice can send another message 3 with a different crypto suite. JFK's SA negotiation uses pre-defined suites.

5.2 Scope of proposals

If a protocol supports different options for different algorithms, the combinations can be stated either as independent choices (if Alice supports three options for encryption and four options for integrity protection, she sends a list of three and a list of four proposals) or as complete suites (in the above case, Alice would send 3*4=12 proposals).

The not-so-hidden agenda in deciding which format to use is trying to avoid the explosion of useless options - all of which should be tested. In the example above, making the choices independent makes the specification and the protocol messages shorter, but it does not reduce the testing effort. Suites are desirable where we expect to *not* support all combinations of algorithms, but only a small subset. If someone demands that the protocol support the new HMAC-SHA2-256 integrity protection algorithm, we would specify suites combining it only with encryption algorithms we would expect to have supported (e.g. AES) rather than all the existing encryption algorithms.

The experience of TLS in this space has been mixed. TLS specifies suites, and the number has grown cumbersome because we were unable to resist requests for lots of questionable combinations. But IKEv1 has its own problems with large numbers of individual choices. In IKEv1, a proposal consisted of a complete set of cryptographic options. This created the opportunity for an exponential explosion of proposals, assuming many of one type of algorithm worked with many of another type of algorithm.

Proposing suites is simpler to encode. The downside of suites are that in practice, it will be less flexible because people will not define suites for every possible combination, and that the specification will be larger because practice with SSL indicates that people will define lots of combinations.

The disadvantage of the flexibility of individual algorithm negotiation is that implementations are likely in practice to support large numbers of combinations without testing them (which works only to the degree that the algorithms don't interact).

Without losing flexibility, IKEv2 allows choices within a proposal, so that if any of several algorithms of type 1 can be used with any of several algorithms of type 2, then the proposal can say "I will do any of these in conjunction with any of those" and the responder selects a single choice for each algorithm. JFK, on the other hand, uses a single suite that is given by the responder as a take-it-or-leave-it offer.

In JFK key agreement, the responder states a single set of individual algorithms that is not negotiable; in JFK SA negotiation, the initiator states a single set of individual algorithms that is not negotiable. In IKEv2, the initiator can propose many options for different algorithms.

5.3 SPD entries

In IKEv2, the TS payload consists of a set of individual traffic selectors. Each selector has a source port; a destination port; and either a range or addresses, a subnet, or a single address. The Responder is allowed to narrow the choices by selecting a subset of the traffic.

In JFK, the initiator proposes an SA; the responder may either accept it or reject it with a reason. The SA offered can have multiple address ranges and multiple port ranges.

6. Wire protocol issues

6.1 Message encoding

IKEv2 uses approximately the same message encoding as IKEv1, using similar payload headers. The main difference is that IKEv2 has a critical bit. IKEv2 adds some new payload types to IKEv1, and uses a completely different encryption and authentication format than was used in IKEv1. It also adds new header flags (initiator and version). The SA payload is also different from IKEv1.

IKEv2 has 4 bit major version number and 4 bit minor version number and cookies to identify the IKE SA and message id to identify the message inside the IKE SA

The JFK message format is simply TLV (tag-length-value) encoding without any generic header. It does not include any kind of version numbers or exchange identifiers.

One feature of the message encoding format that affects speed of execution on many platforms is whether or not the messages and parts of messages align on two- and four-octet boundaries.

6.2 Port number

When a new version of a protocol occurs, the reasons for keeping the same port and simply changing the version number are:

- to avoid the necessity of obtaining a new port

- to avoid the necessity for configuring firewalls to treat the new port like it treated the old port

- to allow implementations of both versions of the protocol in a single software module to send and listen on a single port

- to avoid timeouts when a system that speaks both the new and the old version initiates to a system that only speaks the old version

The reasons to get a new port number for a new version of a protocol are:

- to eliminate the possibility that implementations of the new protocol will interact badly (such as causing a crash) with buggy or not-forward-looking implementations of the old one

- to give the new version total freedom in its choice of wire formats (for example, it is not constrained to put a "version number" in the same place as the old protocol)

- to allow implementations of both versions of the protocol in separate software modules to avoid having to negotiate ownership of the port

JFK proposes assignment of a new port and a wire format that does not place version numbers in compatible places. It does not specify a way for future versions of the protocol to share the new port (assuming that a future version would get another new port).

IKEv2 proposes reusing port 500 and explicitly planning for future versions of the protocol to also share port 500.

6.3 Future versions of the protocols

IKEv2 specifies a method for demanding/requesting that a peer use a particular version of IKE. JFK has no version number and therefore no way of demanding/requesting a particular version.

6.4 Code-preservingness

IKEv2 is clearly based on IKEv1 so a great deal of IKEv1 code can be reused when creating an IKEv2 implementation. Note, however, that IKEv2 changes many parts of IKEv1, so the code reuse must be done very carefully. In a reasonably modular code-base, this might be fairly straightforward. A great deal of torture testing of the new code will have to be done to make sure that the large removals of features no

longer in IKEv2 (such as aggressive mode) don't have unexpected
side-effects.

JFK doesn't preserve any code from IKEv1.

6.5 Extensibility of the protocols

IKEv2 allows private extensions and mechanisms (in the form of critical
and non-critical extensions and version numbers) for future versions of
the protocol to interoperate with the current version.

JFK has no extension mechanism and thus no notion of critical
extensions.

7. References

[CK02] Canetti and Krawczyk, "Security Analysis of IKE's Signature-based
Key-Exchange Protocol", Manuscript, 2002.

[PK00] Perlman and Kaufman, "Key Exchange in IPsec: Analysis of IKE",
IEEE Internet Computing Journal special issue on security solutions, Nov
2000.

8. Summary of features

8.2. Cryptographic features

8.2.1 Identity protection: who is protected, kinds of attacks
- No passive attack, allow active attack against initiator
- Passive attack against responder, no active attack against initiator
- No passive attack, allow active attack against responder

8.2.2 Perfect forward secrecy (PFS)
- Do PFS only when desired (to reduce computational overhead)
- Always do PFS

8.2.3 Authentication styles
- Certificates
- Shared secrets
- Legacy authentication
- Combination of the above

8.2.4 Number of crypto operations
- Different protocols require different numbers of different types of
  crypto operations to set up IPsec SAs and to rekey existing IPsec SAs

8.2.5 Plausible deniability
- Some protocols can make an exchange non-repudiable, others cannot

8.2.6 Formal proofs of security

- Some protocols have formal proofs of security, others do not

8.3. Protocol mechanics

8.3.1 DoS protection

- Protect from clogging DoS attacks by verifying the initiator's existence with an extra round trip
- Protect from clogging DoS attacks by forcing repetition of material in messages 3 and 4 while avoiding the "cookie jar attack"
- Protect from UDP fragmentation attack by having lower-level system check for a value in the ip_id field of the UDP packets
- Protect from UDP fragmentation attack by having lower-level system only accept UDP fragments from IP addresses from which it has received a valid cookie

8.3.2 Number of messages in all scenarios

- Different protocols require different numbers of messages in different scenarios

8.3.3 Size of messages

- Messages are fairly small unless they contain certificates

8.3.4 Preferred ID for responder

- Give an indication by the initiator to the responder as to what identity the responder should use to authenticate to the initiator
- Don't do this

8.3.5 Birth certificates

- Let the other party know that you have crashed and restarted, forgetting all current SAs
- Don't do this

8.4. One or two phases

8.4.1 Control channel vs. separate protocols

- Have a second phase that is an explicit control channel for IPsec SAs
- Control IPsec SAs through other mechanisms

8.4.2 Creating multiple SAs for a single pair of entities

- Create multiple IPsec SAs using only one Diffie-Hellman exchange and one authentication
- Require a Diffie-Hellman exchange and authentication for each IPsec SA created

8.4.3 Dead peer detection

- Look at the last time the SA was used, and possibly send test traffic over the SA if it has timed out
- Send an authenticated message in the SA and wait for the mandatory response

8.4.4 SA deletion

- Negotiate a new SA with an indicator that the SA cannot be used for
  IPsec traffic
- Send a delete notification

8.4.5 SA rekeying
- Replace an old SA with a new one that has different keys

8.4.6 Authenticated error messages
- Protect error messages during SA setup when possible

8.4.7 Authenticated informational messages
- Allow informational messages to be authenticated when possible
- Don't specify informational messages

8.5. SA creation style

8.5.1 Cryptographic agreement

8.5.1.1 Agreement of key exchange keys
- Offer many options and allow the other party to pick the options
  it likes best
- Offer only one option and fail if the other side doesn't like it

8.5.1.2 Agreement of IPsec SA cryptographic algorithms
- Offer many options and allow the other party to pick the options
  it likes best
- Offer only one option and renegotiate if the other side doesn't like
  it

8.5.2 Scope of proposals
- Offer multiple defined suites
- Offer sets of individual options

8.5.3 SPD entries
- Offer multiple selectors, with each selector has a source port; a
  destination port; and either a range or addresses, a subnet, or a
  single address
- Offer a single selector, with the selector having multiple address
  ranges and multiple port ranges

8.6. Wire protocol issues

8.6.1 Message encoding
- Encode similar to IKEv1
- Use tag-length-value encoding

8.6.2 Port number
- Reuse port 500 from IKEv1
- Use a different port

8.6.3 Future versions of the protocols
- Specify version numbers to allow upgrading of versions

- Only allow one version

8.6.4 Code-preservingness

- Base the protocol on IKEv1
- Create an entirely new protocol

8.6.5 Extensibility of the protocols

- Allow extensions
- Allow extensions and allow some of them to be marked as critical
- Don't allow extensions

9. Features in the IKE requirements document

draft-ietf-ipsec-sonofike-rqts-00.txt describes the requirements for the successor to IKE. A large part of that document is a description of the scenarios in which IKE is used. The scenarios highlight some of the items to be considered during protocol design.

The highlights from that document are quoted here linked to the relevant features from above. The highlights appear bracketed with "<<< >>>", and the links to the features in this document are bracketed with "[[[ ]]]". This section assumes that the reader has read draft-ietf-ipsec-sonofike-rqts-00.txt and is matching the text here from the scenarios there.

9.1 Virtual Private Network Site-to-Site Tunnels

<<<This calls out a need for either a two-phased approach for SOI, or a single-phased approach that is sufficiently fast, where "fast" represents an optimal combination of "number of messages" and "computational expenditure".>>> [[[4.1]]]

<<<The cost of authentication must also be factored into the total cost; this will be different for different mechanisms, which results in a decision of scalability -
                      - vs- processing overhead. In certain cases, it may be desirable to amortize the cost of the key management across multiple tunnels.>>> [[[4.2]]]

<<<In the case of a pair of SGWs fronting multiple non-contiguous subnets, a mechanism that allowed the negotiation of a list of phase 2 identities will help to alleviate the number of IPsec tunnels that must be created.>>> [[[5.3]]]

<<<Extensions to the IPsec (now known as phase 2) parameters are needed in order to negotiate QoS characteristics for the various tunnels.>>> [[[6.5]]]

<<<QoS increases the probability of multiple tunnels between a pair of SGWs. Also, negotiation of IPsec tunnels needs to accommodate QoS information, predominantly in the set of selectors used to identify the

contents of any particular IPsec tunnel.>>> [[[4.2]]]

<<<In the case of dynamic IP addresses and/or NAT boxes, IP addresses cannot be used as phase 1 identifiers. This also means that the authentication material cannot be chosen based upon the IP address in the packet.>>> [[[2.3]]]

<<<In a multiple administrative domain network, identity protection of both IPsec endpoints is important.>>> [[[2.1]]]

9.2 Secure Remote Access

<<<This means that the key management mechanism must be able to rapidly establish both the SOI and IPsec connections, without undue impact on CPU and memory overhead. It's also desirable for each exchange to have as few messages as possible, to help alleviate the burst load on the RAS.>>> [[[3]]]

<<<While this does not mandate user authentication to happen within the SOI exchange, it's strongly encouraged that the protocol directly or indirectly associate a single user authentication exchange with a group of IPsec tunnels between a client and an RAS.>>> [[[4.2]]]

<<<For example, this may mean that SOI will need to allow for the client to present its identity (or some "blob of bits" that the server can correctly map to an identity) early in the exchange.>>> [[[4.2]]]

<<<The protocol design (e.g. state management) needs to consider rapid disconnection/reconnection events, where one client disconnects and gives up its (outside, or phase 1) IP address, and the next client grabs that same address.>>> [[[none]]]

<<<Extensions to the IPsec (now known as phase 2) parameters are needed in order to negotiate QoS characteristics for the various tunnels.>>> [[[6.5]]]

<<<QoS increases the probability of multiple tunnels between a pair of SGWs. Also, negotiation of IPsec tunnels needs to accommodate QoS information, predominantly in the set of selectors used to identify the contents of any particular IPsec tunnel.>>> [[[4.2]]]

<<<This means that there must be a way of securely pushing down this policy information before the IPsec tunnel is constructed.>>> [[[2.3]]]

<<<The mechanism associated with such authentication should accommodate re-authentication based on the RAS or authentication server site security policy>>> [[[2.3]]]

<<<Out-of-band authentication mechanisms must also consider the location of the authentication server relative to the client and the RAS. In many scenarios, server tends to be "behind" the RAS device, in the domain that is being secured by the RAS, which may be problematic for

bootstrapping the user authentication for the client-to-RAS connection.>>> [[[2.3]]]

9.3 End-to-End Security

<<<Extensions to the IPsec (now known as phase 2) parameters are needed in order to negotiate QoS characteristics for the various tunnels.>>> [[[6.5]]]

<<<QoS increases the probability of multiple tunnels between a pair of SGWs. Also, negotiation of IPsec tunnels needs to accommodate QoS information, predominantly in the set of selectors used to identify the contents of any particular IPsec tunnel.>>> [[[4.2]]]

<<<Hiding the identity of both parties in the exchange is desirable.>>> [[[2.1]]]

9.4 IP Storage

<<<[ietf-ips-security-xx.txt] discusses resource constraints, calling out the size for both code footprint and data as the most important criteria.>>> [[[6]]]

<<<However, the discussion in [ietf-ips-security-xx.txt] calls out requirements for an API, in order to provide a means of pushing authentication information to the application (e.g. "this peer was authenticated with this cert"), so the application can decide what types of transactions are allowed by this peer.>>> [[[6.5]]]

9.5 PPVPN/MPLS

<<<it may make sense to expand the set of phase 2 identifiers to also support an MPLS/VPN identifier (so the entity doing the SPD check can be separated from the entity doing the encapsulation).>>> [[[6.5]]]

10. Acknowledgements

Most of the material in this document comes from many of the authors of the JFK and IKEv2 proposals, and from the current versions of the proposals themselves. Although many of these people reviewed earlier versions of this document, all errors in the document are the responsibility of the editor.

11. Editor's Address

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060 USA
paul.hoffman@vpnc.org