IPSEC Working Group
INTERNET-DRAFT
draft-ieft-ipsec-ike-01.txt

Dan Harkins, Network Alchemy
Dave Carrel, Redback Networks
May 1999

The Internet Key Exchange (IKE)
<draft-ietf-ipsec-ike-01.txt>

Status of this Memo

Table of Contents

1. Abstract

This memo describes a key exchange and security negotiation protocol which is intended to depricate [HC98]. As such it will not change the "bits on the wire" for an implementation which is compiant with [HC98] but will clarify contentious issues with [HC98] and attempt to explain the protocol in a less haphazard manner. Due to advances in computer processing some mandatory-to-implement attributes have changed between this [HC98] and this document. In addition a new and optional exchange is introduced. Like [HC98] this memo uses [MSST98] for a framework and as a language to express exchanges which are derived from [Kra96] and [Orm98].

In places where the requirements between this document and [MSST98] or [Kra96] or [Orm98] conflict, this document will be supreme.

This is a request/response type protocol in which roles of an Initiator and Responder are played by the parties to the protocol. The Initiator initiates the protocol to the Responder who responds back.

2. Terms and Definitions

2.1 Requirements Terminology

Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in [Bra97].

2.2 Perfect Forward Secrecy

When used in the memo Perfect Forward Secrecy (PFS) refers to the notion that compromise of a single key will permit access to only data protected by a single key. For PFS to exist the key used to protect transmission of data MUST NOT be used to derive any additional keys, and if the key used to protect transmission of data was derived from some other keying material, that material MUST NOT be used to derive any more keys.

Perfect Forward Secrecy for both keys and identities is provided in this protocol.

2.3 Modes and Phases

This protocol uses ISAKMP as a means of communication and all rules regarding transport, construction, and retransmission of messages are as specified in [MSST98]. All messages in IKE are constructed by chaining ISAKMP payloads to an ISAKMP header.

This is a dual phase protocol where the parties to the protocol, also called the peers, authenticate each other and establish a protected communications channel in the first phase and then negotiate security services in the second phase (using the protected communications channel from the first for security).

There are different exchanges, also called "modes", which can be used in the different phases. Phase 1 exchanges are Main Mode and Aggressive Mode. The only phase 2 exchange is Quick Mode. In addition, three additional modes are defined which are neither strictly phase 1 nor phase 2. Section 6 discusses IKE phases in detail.

Unless otherwise noted there are no ordering requirements on payloads

in various IKE messages.

2.4 Security Association

A security association (SA) is a set of policy and key(s) used to protect information. The Phase 1 SA is the shared policy and key(s) used by the negotiating peers in this protocol to protect their communication.

The Phase 1 SA is bi-directional. That is, once established, the peers may use it to securely initiate communication with each other regardless of who initiated the Phase 1 exchange. The roles played by the peers are strictly adhered to during Phase 1 (the Initiator remains the Initiator and the Responder remains the Responder for the entire mode) but once the Phase 1 SA is established the roles can reverse for Phase 2. The peer who was the Responder during Phase 1 can become the Initiator of Phase 2.

Phase 1 SAs are identified by the cookies contained in the ISAKMP header. There are two cookies in the ISAKMP header, one for the Initiator and one for the Responder. The roles during Phase 1 dictate who generates which cookie and once established the cookie order does not swap even if the direction of the Phase 1 SA switches. That is, if Alice initiates a Phase 1 exchange to Bob which results in a Phase 1 SA, the SA will be identified by an Initiator cookie generated by Alice and a Responder cookie generated by Bob even if Bob susequently assumes the role of Initiator during a Phase 2 exchange.

Security Associations are negotiated for other security services during Phase 2. The nature of these SAs is dependant on the Domain of Interpretation (DOI) in the SA payload. ([Pip98] defines the DOI for IPSec).

2.5 Authentication Options

During Phase 1 the peers authenticate each other. There are four distinct methods of authentication in IKE: authentication with pre-shared keys, authentication with digital signatures, and two methods of authentication using public key encryption. These options impact the Phase 1 exchange in a different manner. In fact, the exchange can morph depending on the method chosen.

2.6 Notation

The following notation is used throughout this memo. All "payloads" are from [MSST98].

HDR is an ISAKMP header. When written as HDR* it indicates message

encryption: all payloads following the header are encrypted with a symmetric session key.

SA is an SA negotiation payload with one or more proposal payloads enclosed, which themselves may contain one or more transform payloads.

SAi_b is the entire body of the SA payload (minus the ISAKMP generic header)-- i.e. the DOI, situation, and all proposals and transforms enclosed-- proposed by the Initiator.

CKY-I and CKY-R are the Initiator's cookie and the Responder's cookie, respectively, from the ISAKMP header.

g^i and g^r are the Diffie-Hellman ([DH]) public values of the Initiator and the Responder respectively. The length of these values MUST be the minimum number of octets required to hold a bitstream whose length is equal to the size of the group-- the size of the prime modulus for MODP groups, or the field size for ECP and EC2N groups (see section 5). This requirement is ensured by pre-pending the value with zero bits up to the next 8-bit boundary if necessary.

g^ir is the Diffie-Hellman secret. When used in IKE as input to functions the length of this value MUST be identical to that of g^i and g^r.

KE is the key exchange payload which contains the public value (g^i or g^r) exchanged in a Diffie-Hellman exchange.

Ni and Nr are the nonce payloads from the Initiator or the Responder, respectively. Nonce lengths MUST be between 8 and 256 bytes inclusive.

ID_x is the identification payload for "x". x can be: "i1" or "r1" for the phase one identities of the Initiator or the Responder, respectively; or, "i2" or "r2" for the optional identities exchanged in phase 2.

<P>_b indicates the body of payload <P>-- the ISAKMP generic payload is not included. For instance, Ni_b would signify the body of the nonce payload-- everything except the generic header-- supplied by the Initiator.

SIG is the signature payload.

CERT is the certificate payload.

CERT_REQ is the certificate request payload.

HASH (and any derivitive such as HASH(2) or HASH_I) is the hash payload.

H(x) is the hash of "x" whose result is a digest.

prf(key, msg) is a keyed pseudo-random function-- often a keyed hash function-- used to generate a deterministic output that appears pseudo-random.

SKEYID is a string derived from secret material known only to the active players in the exchange.

SKEYID_e is the keying material used by the parties of the exchange to protect the confidentiality of messages.

SKEYID_a is the keying material used by the parties of the exchange for message integrity.

SKEYID_d is the keying material used to derive keys for security services during phase 2.

<x>y indicates that "x" is encrypted with the key "y".

--> signifies a message from the Initiator to the Responder.

<-- signifies a message from the Responder to the Initiator.

| signifies concatenation of information. For example, X | Y is the concatenation of X with Y.

[x] indicates that x is optional.

3. Introduction

A Domain Of Interpretation defines the attributes (and their meanings) negotiated by IKE. It may overload payloads that are defined in [MSST98]. This protocol does not define its own DOI per se. It does not define DOI nor Situation values for the SA payload during Phase 1 negotiation.

The Phase 1 SA MAY use the DOI and situation from a non-ISAKMP service (such as [Pip97]). In this case an implementation MAY choose to restrict use of the Phase 1 SA for establishment of SAs for services of the same DOI. Alternatively, a Phase 1 SA MAY be established with the value zero in both the DOI and Situation fields and in this case implementations will be free to establish security

services for any defined DOI using this Phase 1 SA. If a DOI of zero is used for establishment of a Phase 1 SA, the syntax of the identity payloads used in the Phase 1 exchange MUST be as defined in [MSST98] and not from any DOI.

3.1 Mandatory Options

Main Mode (section 6.1.1) MUST be implemented for Phase 1; Aggressive Mode (section 6.1.2) SHOULD be implemented. Quick Mode (section 6.2) MUST be implemented for Phase 2. Both acknowledged and unacknowledged notification exchanges (section 6.4) MUST be implemented. New Group mode (section 6.3) SHOULD be implemented.

[MSST98] defines the concept of a "protection suite". To IKE this is the attributes negotiated in SA payloads during Phase 1 whose agreement results in the Phase 1 SA. Attributes that MUST comprise a "protection suite" are:

- encryption algorithm

- hash algorithm

- authentication method

- information about a group over which to do a Diffie-Hellman.

In addition, optional attributes may be negotiated. These include a lifetime and a pseudo-random function ("prf"). (There are currently no negotiable pseudo-random functions defined in this document but the ability to negotiate them exists). In the absense of a negotiated "prf" the HMAC version of the negotiated hash function (see [KCB96]) is used as a pseudo-random function.

Negotiable Phase 1 attributes are described in Appendix A.

IKE implementations MUST support the following attribute values:

- Triple DES in CBC mode for encryption algorithm.

- MD5 [MD5] and SHA [SHA]) for hash function.

- Authentication via pre-shared keys.

- MODP over group number two (see section 5).

In addtion IKE implementations SHOULD support the following values:

- CAST in CBC mode and Blowfish in CBC mode for encryption

algorithm.

- Tiger ([Tiger]) for hash algorithm.

- Authentication using RSA and DSA signatures, and using RSA and El-Gamal public key encryption.

- Groups 3 through 5 for Diffie-Hellman group.

In addition IKE implementations MAY support the following values:

- DES in CBC mode for encryption algorithm.

- Group 1 for Diffie-Hellman group.

3.2 Attribute Negotiation

During security association negotiation Initiators present offers, in the form of protection suites, to Responders. Responders MUST NOT modify any attributes of any offer (with the possible exception of attribute encoding, see Appendix A). If the Initiator of an exchange notices that attribute values have been changed or attributes have been added or deleted from an offer made that response MUST be rejected.

The SA payload from [MSST98] is used to negotiate security associations in both Phase 1 and Phase 2 exchanges. This payload contains a field for a Security Parameter Index (SPI) and a field for the length of the SPI. During Phase 1 the SPI field MUST be empty and the length of the SPI MUST be zero. During Phase 2 the SPI values and their lengths depends on the particular DOI being negotiated.

Diffie-Hellman groups are specified either using a defined group description (section 5) or by defining all attributes of a group (Appendix A) in a protection suite. Group attributes, such as group type or prime number MUST NOT be offered in conjunction with a previously defined group, defined either in section 5 or via a previous New Group Mode exchange (section 6.3).

Certain negotiable attributes have ranges or multiple acceptable values. For instance, if the policy specification on a peer mandates group 2 but is offered group 5, as part of an otherwise acceptable protection suite, the peer SHOULD accept that value as it provides more security than demanded. SA lifetimes pose similar issues. If a peer has a local policy which requires SAs live for no more than 2 hours and is offered a protection suite which contains a lifetime value of 1 hour, the peer SHOULD accept that value as it provides less opportunity for key exposure.

The converse, though, does not hold. If a peer mandates group 5 (or a lifetime of 1 hour) and is offered group 2 (or a lifetime of 1 hour) that offer SHOULD be refused as it violates the local policy.

It is therefore possible to be in a situation where Alice can successfully initiate an IKE exchange with Bob but not the other way around. A simple way around this situation is to not enforce local policy and accept any lifetime offered or any group offered. This behavior is strongly discouraged; implementations SHOULD NOT ignore local policy. If an implementation accepts a protection suite all values of that protection suite MUST be honored-- in other words, implementations MUST NOT ignore lifetime or Diffie-Hellman group offers and just "do their own thing".

A DOI may dictate other actions to take in these circumstances when negotiating its service.

4. IKE Internal State

During an IKE exchange the peers generate state associated with the exchange. This state is generated as soon as all components are available.

4.1 Phase 1 state

Information exchanged during Phase 1 allows for the construction of a Phase 1 SA. This information is protection suite offer(s), cookies (CKY-I and CKY-R), nonces (Ni and Nr), and Diffie-Hellman values (g^i and g^r). In addition to (and out of) this information transmitted in messages some Phase 1 state is generated by each peer. These are the keys the peers use to protect their communications (SKEYID and its derivitives) and the digests they use to authenticate each other.

When the Diffie-Hellman shared secret is used in Phase 1 state generation its length MUST be the minimum number of octets required to hold a bitstream whose length is equal to the size of the group-- the size of the prime modulus for MODP groups, or the field element size for ECP and EC2N groups (section 5). This can be ensured by pre-pending the value with zero bits up to the next 8-bit boundary if necessary.

SKEYID takes different values depending on the authentication method negotiated. The methods of generation are:

digital signatures: SKEYID = prf(Ni_b | Nr_b, g^ir)

public key encryption: SKEYID = prf(H(Ni_b | Nr_b), CKY-I | CKY-R)

pre-shared keys: SKEYID = prf(pre-shared-key, Ni_b | Nr_b)

Upon generation of SKEYID, the remaining internal state can be derived. SKEYID_d is used to derive additional keys for security services other than IKE during a Phase 2 exchange; SKEYID_a is used to provide authentication and data integrity to IKE messages; and, SKEYID_e is used to provide confidentiality to IKE messages.

SKEYID_d = prf(SKEYID, g^ir | CKY-I | CKY-R | 0)

SKEYID_a = prf(SKEYID, SKEYID_d | g^ir | CKY-I | CKY-R | 1)

SKEYID_e = prf(SKEYID, SKEYID_a | g^ir | CKY-I | CKY-R | 2)

where 0, 1, and 2 are represented by a single octet.

As part of the authentication step each side generates two digests, one for the Initiator, I, and one for the Responder, R. These digests either presented as is or digitally signed depending on the authentication method negotiated. These digests are:

I-digest = prf(SKEYID, g^i | g^r | CKY-I | CKY-R | SAi_b | ID_i1_b)

R-digest = prf(SKEYID, g^r | g^i | CKY-R | CKY-I | SAi_b | ID_r1_b)

For authentication with digital signatures, I and R are digitally signed and the resulting signature is passed as a SIG payload during the exchange. For authentication with pre-shared keys and both types of public key encryption I and R are passed as a HASH payload during the exchange.

Symmetric encryption algorithms used by IKE to protect its messages MUST be in CBC mode. This mode requires an Initialization Vector (IV) for each encrytion operation. The initial IV is derived from a hash of the concatenation of the Initiator's Diffie-Hellman public value and the Responder's Diffie-Hellman public value using the negotiated hash algorithm. The public values are those taken directly out of the KE payload including any pre-pended zero bits. This is used for the first message only. Subsequent messages MUST use the last CBC encryption block from the previous message as their initialization vector.

4.2 Phase 2 state

During a Quick Mode exchange new SAs for a security service other than ISAKMP are generated (e.g. IPSec). The exact nature of those SAs is defined in the approprate DOI document, but as part of the key generation aspect of Quick Mode negotiation IKE does require some new

state be maintained.

Quick Mode exchanges are done under the protection of an existing Phase 1 exchange and the message ID from the ISAKMP header is used to multiplex multiple exchanges through the single SA. While the Phase 1 SA is identified by the cookies during a Phase 1 exchange, it is the message ID and the cookies that identifies a state specific to a Phase 2 exchange.

Unlike a Phase 1 exchange there are no variables that need to be calculated and retained after the exchange has finished. Only information exchanged during a Quick Mode-- protection suite offers, nonces and, if PFS is desired, additional Diffie-Hellman public values-- need be maintained and once the exchange has terminated it can be thrown away (after, of course, the key(s) for the SA(s) have been supplied to the appropriate service). The actual method for generating keying material is discussed in section 6.2.

All Quick Mode messages are encrypted using the negotiated symmetric cipher with SKEYID_e as the key. Therefore a seperate IV is needed to "jump start" the encryption. This IV is derived from a hash of the concatenation of the last Phase 1 CBC output block (or the Phase 1 IV derived in section 4.1 if no Phase 1 messages were encrypted), and the Phase 2 message ID using the negotiated hash algorithm.

5 Oakley Groups

There are 5 groups different Diffie-Hellman groups defined for use in IKE. These groups were generated by Richard Schroeppel at the University of Arizona. Properties of these primes are described in [Orm96].

The strength supplied by group one may not be sufficient for the mandatory-to-implement encryption algorithm and is here for historic reasons.

5.1 First Oakley Group

IKE implementations MAY support a MODP group with the following prime and generator. This group is assigned id 1 (one).

The prime is: 2^768 - 2 ^704 - 1 + 2^64 * { [2^638 pi] + 149686 }
Its hexadecimal value is

FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08
8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B
302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9
A63A3620 FFFFFFFF FFFFFFFF

The generator is: 2.

5.2 Second Oakley Group

IKE implementations MUST support a MODP group with the following prime and generator. This group is assigned id 2 (two).

The prime is 2^1024 - 2^960 - 1 + 2^64 * { [2^894 pi] + 129093 }. Its hexadecimal value is

FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08
8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B
302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9
A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6
49286651 ECE65381 FFFFFFFF FFFFFFFF

The generator is 2 (decimal)

5.3 Third Oakley Group

IKE implementations SHOULD support a EC2N group with the following characteristics. This group is assigned id 3 (three). The curve is based on the Galois Field GF[2^155]. The field size is 155. The irreducible polynomial for the field is:

u^155 + u^62 + 1.

The equation for the elliptic curve is:

y^2 + xy = x^3 + ax^2 + b.

Field Size: 155
Group Prime/Irreducible Polynomial:
0x08000000000000000000000040000000000000001
Group Generator One: 0x7b
Group Curve A: 0x0
Group Curve B: 0x07338f
Group Order: 0x08000000000000000057db5698537193aef944

The data in the KE payload when using this group is the value x from the solution (x,y), the point on the curve chosen by taking the randomly chosen secret Ka and computing Ka*P, where * is the repetition of the group addition and double operations, P is the curve point with x coordinate equal to generator 1 and the y coordinate determined from the defining equation. The equation of curve is implicitly known by the Group Type and the A and B coefficients. There are two possible values for the y coordinate; either one can be used successfully (the two parties need not agree on the selection).

5.4 Fourth Oakley Group

IKE implementations SHOULD support a EC2N group with the following characteristics. This group is assigned id 4 (four). The curve is based on the Galois Field GF[2^185]. The field size is 185. The irreducible polynomial for the field is:

u^185 + u^69 + 1.

The equation for the elliptic curve is:

y^2 + xy = x^3 + ax^2 + b.

Field Size: 185
Group Prime/Irreducible Polynomial:
0x020000000000000000000000000002000000000000000001
Group Generator One: 0x18
Group Curve A: 0x0
Group Curve B: 0x1ee9
Group Order: 0x01ffffffffffffffffffffdbf2f889b73e484175f94ebc

The data in the KE payload when using this group will be identical to that as when using Oakley Group 3 (three).

5.5 Fifth Oakley Group

IKE implementations SHOULD support a MODP group with the following prime and generator. This group is assigned id 5 (five).

The prime is 2^1536 - 2^1472 - 1 + 2^64 * {[2^1406 pi] + 741804}. Its hexadecimal value is

FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08
8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B
302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9
A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6
49286651 ECE45B3D C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8
FD24CF5F 83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA237327 FFFFFFFF FFFFFFFF

The generator is 2.

Other groups can be defined using New Group Mode.

6. Modes for IKE Phases

IKE exchanges are called modes and these modes accomplish the phased negotiation for IKE. There are two phases. Phase 1 exchanges are intended to establish shared policy and keys in the form of a Phase 1 Security Association (SA). Phase 2 exchanges are intented to establish Security Associations for security services other than IKE.

A DOI defines the nature of those services. [Pip98] is the DOI for IPSec.

Exchanges in IKE are not open-ended and a certificate request payload MUST NOT extend the number of messages in a given exchange.

6.1 Phase 1

There are three steps to Phase 1: negotiation of protection suites, a Diffie-Hellman exchange, authentication. There are two modes to accomplish Phase 1: Main Mode, and Aggressive Mode. Their most obvious difference is in the number of message necessary to accomplish the work. Main Mode requires 6 messages while Aggressive Mode requires 3.

There are other subtle differences. For instance, the parameters to negotiate in an Aggressive Mode exchange are constrained compared to a Main Mode exchange. Because the Diffie-Hellman exchange begins with the first message (which also contains the protection suite offers) it is not possible to negotiate offers with differing Diffie-Hellman group attributes. Other parameters are constrained depending on the authentication method chosen.

All initial Phase 1 messages from the Initiator to the Responder MUST have an SA payload as the first payload following the ISAKMP header. The SA payload MUST contain only one proposal payload. Multiple protection suites are offered by offering multiple transform payloads and encapsulating them in the single SA payload. There is no limit on the number of offers an Initiator can make but compliant implementations MAY, for performance reasons,
choose to limit the number of offers it will inspect.

Phase 1 authentication using Public Key Encryption requires the Initiator to possess the Responder's public key prior to initiation of the exchange. The method of acquisition of the Responder's public key is outside the scope of this memo. It can be any out-of-band mechanism or it can be from a previous IKE exchange in which certificates were requested, exchanged, and retained. If a previous exchange requests a peer's certificate for subsequent use during an exchange using public key encryption as the authentication method the certificate encoding requested MUST be for key exchange (5).

Use of the commit bit from [MSST98] during Phase 1 is forbidden. Implementations SHOULD respond with an notify message whose type is set to INVALID-FLAGS (8).

6.1.1 Main Mode

A goal of Main Mode is identity protection. The ID payloads are never passed on the wire in the clear, thereby masking the true identity of the parties performing the IKE exchange. Main Mode is an instantiation of the ISAKMP Identity Protect exchange from [MSST98].

6.1.1.1 Main Mode Authentication With Pre-Shared Keys

A key derived from some out-of-band mechanism (which is beyond the scope of this memo) can be used to authenticate an IKE exchange. When using pre-shared key authentication, Main Mode is defined as:

```
    Initiator                    Responder
   -----------                  -----------
    HDR, SA             -->
                        <--     HDR, SA
    HDR, KE, Ni         -->
                        <--     HDR, KE, Nr
    HDR*, IDi1, HASH_I  -->
                        <--     HDR*, IDr1, HASH_R
```

Where HASH_I and HASH_R are the I-digest and R-digest (section 4.1), respectively, presented in a hash payload.

When using pre-shared key authentication with Main Mode the key can only be identified by the IP address of the peers since computation of the I-digest is dependant on the pre-shared key and I-digest must be computed prior to the Responder receiving IDi1.

6.1.1.2 Main Mode Authentication with Digital Signatures

Digital signatures may be used to authenticate a Main Mode exchange. Signature checking requires trusting a public key and, in lieu of a Public Key Infrastructure, certificates can be passed in-line to faccilitate this. Main Mode authenticated with digital signatures is defined as:

```
    Initiator                          Responder
   -----------                        -----------
    HDR, SA                     -->
                                <--    HDR, SA
    HDR, KE, Ni                 -->
                                <--    HDR, KE, Nr
    HDR*, IDi1, [ CERT, ] SIG_I -->
                                <--    HDR*, IDr1, [ CERT, ] SIG_R
```

Where SIG_I and SIG_R are digital signatures of I-digest and R-digest (section 4.1), respectively, and presented in a signature payload.

Since the algorithm used in generation of I-digest and R-digest is known (it is the prf which is, most likely, the HMAC version of the negotiated hash algorithm) there is no need to encode its OID into the signature. In addition, there is no binding between the OIDs used for RSA signatures in [PKCS1] and those used in this document. Therefore, RSA signatures MUST be encoded as a private key encryption in [PKCS1] format (without the hash OID) and not as a PKCS #1 signature (which encodes the digest with an OID). DSA signatures MUST be encoded as the value "r" followed by "s".

In general the signature will be directly over I-digest and R-digest which are generated by the pseudo-random function. However, this can be overridden for construction of a signature if the signature algorithm is tied to a particular hash algorithm by changing the digest construction from:

```
digest = prf(key, msg)
```

to

```
digest = hash(key | msg)
```

For example, DSS is only defined with SHA's 160 bit output and in this case the digests would be:

```
I-digest = SHA(SKEYID | g^i | g^r | CKY-I | CKY-R | SAi_b |
               ID_i1_b)

R-digest = SHA(SKEYID | g^r | g^i | CKY-R | CKY-I | SAi_b |
               ID_r1_b)
```

The contents of the signature payload would then be the resulting 320-bit DSA signature of the digest ("r" followed by "s").

6.1.1.3 Main Mode Authentication with Public Key Encryption

Main Mode can be authenticated by using public key encryption by encrypting each nonce in the peer's public key. The ability of the peer to decrypt the nonce and properly construct the authenticating digest from section 4.1 is authenticated proof of identity. This authentication method is attractive in that it is a deniable exchange. The peers authenticate each other but they lack the kind of non-repudiable proof of conversation that one gets with a digital signature. In addition, security is added to secret generation since an attack would have to successfully break not only the Diffie-

Hellman exchange but also both public key encryptions. This exchange was adapted from [SKEME].

In order to perform public key encryption authentication the Initiator must already have the Responder's public key. In the case where the Responder has multiple public keys, a hash of the certificate which contains the pubic key the Initiator is using to encrypt the nonce is passed as part of the third message. When passed, the hash payload MUST precede any payloads encrypted with the Responder's public key. The Responder identifies the Initiator's public key using the Inititator's identity passed as IDi1.

The Nonce and ID payloads MUST be encrypted in the public key of the peer.

Main Mode authenticated with public key encryption is defined as:

```
    Initiator                          Responder
   -----------                        -----------
    HDR, SA                    -->
                               <--     HDR, SA
    HDR, KE, [ HASH(1), ]
       <IDi1_b>PubKey_r,
        <Ni_b>PubKey_r         -->
                                       HDR, KE, <IDr1_b>PubKey_i,
                               <--             <Nr_b>PubKey_i
    HDR*, HASH_I               -->
                               <--     HDR*, HASH_R
```

Where HASH(1) is the optional hash of the certificate which contained Pubkey_r. Where HASH_I and HASH_R are the I-digest and R-digest, respectively, from section 4.1 presented in a hash payload.

The format of the encrypted data depends on the algorithm being used. For RSA encryption the format is specified in [PKCS1]. For El-Gamal encryption the [PKCS1] convention for encryption block encoding is used for each component of the ciphertext, and the ciphertext of message M consists of the value A followed by the value B:

```
A = g^k mod p
B = y^k * M mod p
```

where the peer's public key is y, g, p and k is a random value per the El-Gamal cryptosystem.

6.1.1.4 Main Mode Authentication with Revised Public Key Encryption

Authentication with public key encryption has significant advantages

over authentication with digital signatures (see 6.1.1.3 above). Unfortunatly, this is at the cost of 4 public key operations-- two public key encryptions and two private key decryptions. The revised mode of public key authentication retains the advantages to public key encryption but does so with half the public key operations.

In this mode the nonce is still encrypted using the public key of the peer, however all subsequent payloads are encrypted using the negotiated symmetric encryption algorithm (from the SA payload) with a key derived from the nonce. This solution adds minimal complexity and state yet saves two costly public key operations on each side. In addition the KE payload is also encrypted using this same derived key to provide additional protection against cryptanalysis of the Diffie-Hellman exchange.

As with the authentication method from 6.1.1.3 a hash payload may be sent to identify a certificate if the Responder has multiple certificates which contain useable public keys. If the hash payload is sent is MUST be the first payload of the third message and MUST be followed by the encrypted nonce. If the hash payload is not sent the first payload MUST be the encrypted nonce. All payloads, including any optional payloads, following the nonce MUST be encrypted in the appropriate symmetric key derived from the encrypted nonce.

Main Mode authenticated with the revised method of public key encryption is defined as:

```
    Initiator                          Responder
   -----------                        -----------
    HDR, SA                    -->
                               <--     HDR, SA

    HDR, [ HASH(1), ]
      <Ni_b>Pubkey_r,
      <KE_b>Ke_i,
      <IDi1_b>Ke_i,
      [<<Cert-I_b>Ke_i]        -->
                                       HDR, <Nr_b>PubKey_i,
                                            <KE_b>Ke_r,
                               <--          <IDr1_b>Ke_r,
    HDR*, HASH_I               -->
                               <--     HDR*, HASH_R
```

Where HASH(1) is the optional hash of the certificate which contained Pubkey_r, HASH_I and HASH_R are the I-digest and R-digest, respectively, from section 4.1 presented in a hash payload, and Ke_i and Ke_r are the keys to the symmetric encryption algorithm negotiated in the SA payload. Note that only the bodies of the payloads are encrypted (in both asymmetric and symmetric operations),

the generic headers are left in the clear to enable proper payload parsing. The payload length includes that added to perform encryption.

The format of the ciphertext depends on the algorithm used and is identical to that of section 6.1.1.3.

The symmetric cipher keys are derived from the decrypted nonces as follows. First the values Ne_i and Ne_r are computed:

```
Ne_i = prf(Ni_b, CKY-I)
Ne_r = prf(Nr_b, CKY-R)
```

The keys Ke_i and Ke_r are then taken from Ne_i and Ne_r respectively in the manner described in Appendix B used to derive symmetric keys for use with the negotiated encryption algorithm. If the length of the output of the negotiated prf is greater than or equal to the key length requirements of the cipher, Ke_i and Ke_r are derived from the most significant bits of Ne_i and Ne_r respectively. If the desired length of Ke_i and Ke_r exceed the length of the output of the prf the necessary number of bits is obtained by repeatedly feeding the results of the prf back into itself and concatenating the result the necessary number has been achieved. For example, if the negotiated encryption algorithm requires 320 bits of key and the output of the prf is only 128 bits, Ke_i is the most significant 320 bits of K, where

```
K = K1 | K2 | K3 and
K1 = prf(Ne_i, 0)
K2 = prf(Ne_i, K1)
K3 = prf(Ne_i, K2)
```

For brevity, only derivation of Ke_i is shown; Ke_r is identical. The length of the value 0 in the computation of K1 is a single octet. Note that Ne_i, Ne_r, Ke_i, and Ke_r are all ephemeral and MUST be discarded after use.

Save the requirements on the location of the optional HASH payload and the mandatory nonce payload there are no further payload requirements. All payloads-- in whatever order-- following the encrypted nonce MUST be encrypted with Ke_i or Ke_r depending on the direction.

CBC mode is used with symmetric encryption in IKE and this requires an IV. The IV for encrypting the first payload following the nonce is set to 0 (zero). The IV for subsequent payloads encrypted with the ephemeral symmetric cipher key, Ke_i, is the last ciphertext block of the previous payload. Encrypted payloads are padded up to the nearest

block size. All padding bytes, except for the last one, contain 0x00. The last byte of the padding contains the number of the padding bytes used, excluding the last one. Note that this means there will always be padding.

6.1.2 Aggressive Mode

An Aggressive Mode exchange completes Phase 1 with half the messages as Main Mode. It does this at some expense of negotiation. Because more payloads are sent with the first two messages more committment to the nature of the exchange must be made. For instance, the KE payload, which contains the Diffie-Hellman public value, is sent in the same message as the SA payload which contains attributes for negotiation of the Diffie-Hellman group. Obviously, it is not possible to negotiate this attribute. Different authentication methods can further constrain the negotiable options of Aggressive Mode.

Unlike Main Mode, Aggressive Mode was not designed for Identity Protection. The ID payloads are passed in the clear allowing a potential adversary to observe the identities of the parties to the IKE exchange.

An advantage of Aggressive Mode over Main Mode (aside from the fewer rounds) is that computation of SKEYID state can be delayed until transmission (and receipt) of the final authenticating message from the Initiator to the Responder. All diagrams show this final Aggressive Mode message as unencrypted (it lacks the asterisk with the HDR). Implementations MAY choose to send this message encrypted though.

6.1.2.1 Aggressive Mode Authentication with Pre-Shared Keys

Because the ID payload is passed in Aggressive Mode prior to computation of SKEYID state it is possible to use pre-shared keys which are bound to identities other than the peer's IP address. For situations where the Initiator is using a dynamically assigned IP address this is especially important. In spite of the fact that the ID payload is passed in the clear an identity can be "hidden" by using the KEY_ID identity type from [MSST98]. This enables a pre-shared key to be associated with an opaque sequence of characters which conveys no meaning to anyone except the two parties to the exchange.

Aggressive Mode authenticated with pre-shared keys is defined as:

Initiator	Responder

```
      HDR, SA, KE, Ni, IDi1 -->
                           <--    HDR, SA, KE, Nr, IDr1, HASH_R
      HDR, HASH_I          -->
```

Where the payload contents are identical to Main Mode.

6.1.2.2 Aggressive Mode Authenticated with Digital Signatures

Digital signatures can authenticate Aggressive Mode in the same manner in which they authenticate Main Mode.

```
    Initiator                          Responder
   -----------                        -----------
    HDR, SA, KE, Ni, IDi1   -->
                            <--    HDR, SA, KE, Nr, IDr1,
                                        [ CERT, ] SIG_R
    HDR, [ CERT, ] SIG_I    -->
```

The optional CERT payload sent by the Initiator is shown as part of the last Aggressive Mode message but since [MSST98] allows for a freeform construction of messages this payload MAY be sent as part of the first message.

6.1.2.3 Aggressive Mode Authenticate with Public Key Encryption

This authentication method further constrains Aggressive Mode authentication in that if this method is desired than all protection suites offered in the SA payload must offer it. Note that Aggressive Mode authenticated with both pre-shared keys and digital signatures may offer multiple protection suites, some with pre-shared key offers, some with digital signature offers because the Initiator has not yet committed himself to an authentication method. This is not the case with authentiction using public key encryption. The nonce payload, which is part of the first message, must be encrypted in the Responder's public key so the Initiator is bound to this method if it is desired.

Like Main Mode authentication with public key encryption, a hash payload may be sent by the Initiator prior to the use of the Responder's public key to identify which public key is being used in the event the Responder has multiple certificates. The Nonce payloads and ID payloads MUST be encrypted in the peer's public key. Also, like Main Mode this is a completely deniable exchange.

An advantage of using this method of authentication in Aggressive Mode is that identity protection can be realized because the ID payloads are similarly encrypted in the peer's public key along with the nonce payload.

Aggressive Mode authenticated with public key encryption is defined as:

```
    Initiator                          Responder
   -----------                        -----------
    HDR, SA, [ HASH(1),] KE,
      <IDi1_b>Pubkey_r,
       <Ni_b>Pubkey_r          -->
                                       HDR, SA, KE, <IDr1_b>PubKey_i,
                               <--          <Nr_b>PubKey_i, HASH_R
    HDR, HASH_I                -->
```

Where the notation, formatting, and payload contents are identitical to section 6.1.1.3.

6.1.2.4 Aggressive Mode Authentication with Revised Public Key Encryption

The revised method of public key encryption affords the same benefits to Aggressive Mode as it did to Main Mode: the exchange is still completely deniable, an adversary must attack not only the Diffie-Hellman exchange but both public key encryptions, and it requires only two public key operations, a public key encryption and a private key decryption.

As with the Revised Public Key Encryption method in Main Mode, a hash of the Responder's public key may be sent prior to the use of the key in the event that the Responder has multiple public keys. Also, all payloads following the encrypted nonce, including any optional payloads, MUST be encrypted with the negotiated symmetric cipher (from the SA payload) using a key derived from the nonce.

The revised method of public key encryption further constrains the negotiable options that the Initiator can offer to the Responder. This is due to the fact that a hash algorithm and a symmetric encryption algorithm must be used to construct the first message. In fact, all mandatory attributes that must accompany all protection suite offers must be identical. The only way to construct multiple protection suite offers using Aggressive Mode with the revised method of public key encryption is to vary the lifetime.

```
      Initiator                          Responder
     -----------                        -----------
      HDR, SA, [ HASH(1),]
        <Ni_b>Pubkey_r,
        <KE_b>Ke_i, <IDii_b>Ke_i
        [, <Cert-I_b>Ke_i ]      -->
                                          HDR, SA, <Nr_b>PubKey_i,
                                               <KE_b>Ke_r, <IDir_b>Ke_r,
                                 <--           HASH_R
      HDR, HASH_I                -->
```

Where notation, formatting, payload contents and all state generation is identical to section 6.1.1.4.

6.2 Quick Mode

A Quick Mode exchange is strictly a Phase 2 exchange and MUST be done under the protection of an existing Phase 1 SA. SKEYID_a is used, in conjunction with the appropriate prf, to authenticate Phase 2 messages and SKEYID_e is used, in conjunction with the negotiated symmetric cipher, to encrypt Phase 2 messages.

As mentioned in section 4.2 the message ID and cookies from the ISAKMP SA identifies the Phase 1 SA and transient Phase 2 state for a Quick Mode.

The first payload, following the ISAKMP header, of a Quick Mode exchange MUST be a hash payload and following it MUST be an SA payload.

Quick Mode is essentially SA negotiation and an exchange of nonces that provides replay protection. The nonces are used to generate fresh keying material and to prevent replay attacks from generating bogus security associations. If PFS is desired optional KE payloads can be exchanged to allow for an additional Diffie-Hellman exchange and exponentiation per Quick Mode. While use of the key exchange payload with Quick Mode is optional it MUST be supported.

The security service for which the Quick Mode exchange is being performed may require identities to be supplied along with the SA and key(s). Those identities are implicitly assumed to be the IP address of the IKE peers, without any implied constraints on the protocol or port numbers allowed, unless additional ID payloads (called "client IDs") are passed. Client IDs come in pairs, an Initiator ID, IDi2, and a Responder ID, IDr2, to denote to the service where the information being secured will come from and to where it is going--e.g. if IKE is negotiating SAs for IPSec on a security gateway the client IDs could specify that the SAs are for traffic from a

particular IP address (IDi2) to a particular IP address (IDr2). The order is crucial and the first ID in any Quick Mode MUST be the IDi2 and the second MUST be IDr2. For parsing sanity's sake, IDr2 MUST immediately follow IDi2.

In the case of a Quick Mode Initiator, these client IDs are be supplied by the service requesting SAs. In the case of a Quick Mode Responder, those IDs are extracted from the Quick Mode message and supplied to the service identified by the DOI value in the SA payload (also in the Quick Mode message). If the identities are not acceptable to the service (due to policy or other reasons), an Informational message containing a notify payload, with a type of INVALID-ID-INFORMATION (18), SHOULD be sent back to the Initiator. The Responder MUST NOT modify the client IDs in any way and they MUST be delivered back to the Initiator in exactly the order supplied, that is, IDi2 does not become IDr2 in the message the Responder sends back to the Initiator.

Client IDs are used to constrain the use of security associations. For example, SAs negotiated under [Pip98] can be have varying granularities by specifying (or not) particular port and protocol combinations along with the ID type. In this fashion, a single pair of SAs can protect all traffic between two subnets, and a separate pair can protect only telnet traffic between two particular hosts.

The DOI of the service being negotiated defines the exact format that client IDs may take.

All offers made during a Quick Mode are logically related and MUST be consistant. For example, if a KE payload is send the attribute describing the Diffie-Hellman group (see section 5 and [Pip98]) MUST be included in every transform of every proposal of every SA being negotiated. Similarly, if client IDs are used they MUST apply to every SA negotiated.

Quick Mode is defined as:

```
    Initiator                          Responder
   -----------                        -----------
    HDR*, HASH(1), SA, Ni
      [, KE ] [, IDi2, IDr2 ]  -->
                                <--    HDR*, HASH(2), SA, Nr
                                             [, KE ] [, IDi2, IDr2 ]
    HDR*, HASH(3)               -->
```

where:

HASH(1) is the results of a prf over the message ID (M-ID) of the

ISAKMP header concatenated with the entire message that follows the hash including all payload headers, but excluding any padding added for encryption.

HASH(2) is identical to HASH(1) except that the Initiator's nonce-- Ni, minus the payload header-- is added after M-ID and prior to the rest of the message. The addition of the nonce to HASH(2) is for a liveliness proof.

HASH(3) is the results of a prf over the value zero represented as a single octet, followed by a concatenation of the message ID and the two nonces-- the Initiator's followed by the Responder's. This is for a liveliness proof.

Graphically, the hashes are:

```
HASH(1) = prf(SKEYID_a, M-ID | SA | Ni [ | KE ]
              [ | IDci | IDcr ] )
HASH(2) = prf(SKEYID_a, M-ID | Ni_b | SA | Nr [ | KE ]
              [ | IDci | IDcr ] )
HASH(3) = prf(SKEYID_a, 0 | M-ID | Ni_b | Nr_b)
```

With the exception of the hash, SA and client ID payloads there are no payload ordering restrictions on Quick Mode. HASH(1) and HASH(2) may differ from the illustration above if the order of payloads in the message differs from the illustrative example or if any optional payloads, for example a notify payload, have been chained to the message.

The commit bit in the ISAKMP header ([MSST98]) can be used to extend a Quick Mode by a single message from the Responder to the Initiator to delay use of the SAs created by the Quick Mode. This message will consist of an authenticated hash, using SKEYID_a as the key, of the message ID from the Quick Mode concatinated with a notify payload whose type is set to CONNECTED (16384). This final message is sent as part of the Quick Mode exchange and not as a seperate Informational exchange. Either side can set the commit during the exchange and the other party SHOULD reflect back, or acknowledgement, the commit bit in a subsequent message. Using the '#' symbol to denote the message with the commit bit, a Quick Mode exchange would become:

```
       Initiator                          Responder
      -----------                        -----------
       HDR*, HASH(1), SA, Ni
         [, KE ] [, IDi2, IDr2 ] -->
                                  <--    HDR*#, HASH(2), SA, Nr
                                               [, KE ] [, IDi2, IDr2 ]
       HDR*#, HASH(3)             -->
                                  <--    HDR#*, HASH(4), notify
```

where HASH(4) = prf(SKEYID_a, M-ID | notify).

If PFS is not desired and KE payloads were not exchanged the keying material generated by IKE to accompany the SA is defined as:

KEYMAT = prf(SKEYID_d, protocol | SPI | Ni_b | Nr_b)

If PFS is desired and KE payloads were exchanged the keying material generated by IKE to accompany the SA is defined as:

KEYMAT = prf(SKEYID_d, g(qm)^xy | protocol | SPI | Ni_b | Nr_b)

where g(qm)^xy is the shared secret from the ephemeral Diffie-Hellman exchange of this Quick Mode.

In either case, "protocol", and "SPI", are from the ISAKMP proposal payload that contained the negotiated (and accepted) transform payload.

A single SA negotiation results in two security associations-- one inbound and one outbound. Different SPIs for each SA (one chosen by the Initiator, the other by the Responder) guarantee a different key for each direction. The SPI chosen by the destination of the SA is used to derive KEYMAT for that SA.

For situations where the amount of keying material desired is greater than that supplied by the prf, KEYMAT is expanded by feeding the results of the prf back into itself and concatenating results until the required keying material has been reached. In other words,

```
KEYMAT = K1 | K2 | K3 | ...
where:
   K1 = prf(SKEYID_d, [ g(qm)^xy | ] protocol | SPI | Ni_b | Nr_b )
   K2 = prf(SKEYID_d, K1 | [ g(qm)^xy | ] protocol | SPI | Ni_b |
            Nr_b )
   K3 = prf(SKEYID_d, K2 | [ g(qm)^xy | ] protocol | SPI | Ni_b |
            Nr_b )
   etc.
```

This keying material (whether with PFS or without, and whether derived directly or through concatenation) MUST be used with the negotiated SA. It is up to the service to define how keys are derived from the keying material.

In the case of an ephemeral Diffie-Hellman exchange in Quick Mode, the exponential (g(qm)^xy) MUST be irretrievably removed from the current state and SKEYID_e and SKEYID_a (derived from the Phase 1 negotiation) continue to protect and authenticate the Phase 1 SA and SKEYID_d continues to be used to derive keys for subsequent Quick Modes.

Using Quick Mode, multiple SA's and keys can be negotiated with one exchange as follows:

```
    Initiator                       Responder
   -----------                     -----------
    HDR*, HASH(1), SA0, SA1, Ni,
      [, KE ] [, IDi2, IDr2 ] -->
                              <--   HDR*, HASH(2), SA0, SA1, Nr,
                                       [, KE ] [, IDi2, IDr2 ]
    HDR*, HASH(3)             -->
```

The keying material is derived identically as in the case of a single SA. In this case (negotiation of two SA payloads) the result would be four security associations-- two each way for both SAs.

6.3 New Group Mode

IKE uses 5 groups from [Orm96] to choose from when doing Diffie-Hellman exchanges. Sometimes those groups may not be desirable for one reason or another. In this case, New Group Mode can be used to establish a new shared group between cooperating peers.

New Group mode uses an SA payload from [MSST98] to convey the attributes of the group being established. This payload contains a field for the Security Parameter Index (SPI) and a field for the SPI length. During a New Group mode exchange the SPI field MUST be empty and its length set to zero.

New Group Mode MUST NOT be used prior to establishment of an IKE SA. The description of a new group MUST only follow phase 1 negotiation. (It is not a phase 2 exchange, though).

```
    Initiator                       Responder
   -----------                     -----------
    HDR*, HASH(1), SA         -->
                              <--   HDR*, HASH(2), SA
```

where HASH(1) is the prf output, using SKEYID_a as the key, and the message-ID from the ISAKMP header concatenated with the entire SA proposal, body and header, as the data; HASH(2) is the prf output, using SKEYID_a as the key, and the message-ID from the ISAKMP header concatenated with the reply as the data. In other words the hashes for the above exchange are:

```
HASH(1) = prf(SKEYID_a, M-ID | SA)
HASH(2) = prf(SKEYID_a, M-ID | SA)
```

The proposal will specify the characteristics of the group (see appendix A, "Attribute Assigned Numbers"). Group descriptions for private Groups MUST be greater than or equal to 2^15. If the group is not acceptable, the responder MUST reply with a Notify payload with the message type set to ATTRIBUTES-NOT-SUPPORTED (13).

IKE implementations MAY require private groups to expire with the SA under which they were established.

Groups may be directly negotiated in the SA proposal with Main Mode. To do this the component parts-- for a MODP group, the type, prime and generator; for a EC2N group the type, the Irreducible Polynomial, Group Generator One, Group Generator Two, Group Curve A, Group Curve B and Group Order-- are passed as SA attributes (see Appendix A). Alternately, the nature of the group can be hidden using New Group Mode and only the group identifier is passed in the clear during phase 1 negotiation.

6.4 Notification Exchanges

At various point during an IKE exchange peers may desire to convey some information to each other regarding errors or notifications of certain state transitions. To accomplish this IKE defines two Informational exchanges, an unreliable uni-directional message and an acknowledged, reliable bi-directional exchange.

Informational exchanges described in this memo are secured by a previously established Phase 1 SA. Any Information message sent prior the the establishment of a Phase 1 SA MUST be sent unsecured to prevent a loss of cryptographic syncronization. Any action taken upon receipt of an unprotected Informational message should be taken with great care as they can easily be forged by any evesdropper. Therefore, IKE implementations are discouraged from sending unprotected Informational messages. Exceptions to this are for Informational messages which will not effect continuation of the exchange. For example, sending an INVALID-FLAGS message for improper use of the commit bit would not necessarily cause termination of an exchange in the way a NO-PROPOSAL-CHOSEN message would.

Like a Quick Mode exchange, Informational messages are given a pseudo-random message ID to allow for multiplexing exchanges through a single Phase 1 SA. Likewise, they are secured by a Phase 1 SA: confidentiality is provided by SKEYID_e and message integrity is provided by a keyed hash using SKEYID_a.

The initiaization vector for these exchanges is derived in exactly the same fashion as that for a Quick Mode-- i.e. it is derived from a hash of a concatenation of the last phase 1 CBC output block (or the Phase 1 IV if no Phase 1 messages were encrypted) and the message id from the ISAKMP header of the Informational Exchange (not the message id from the message that may have prompted the Informational Exchange).

Due to the freeform nature of ISAKMP message construction more than one single notify payload may be sent in a single Informational exchange.

6.4.1 Unacknowledged Informational

An unacknowledged, FYI-style, Informational message is defined as:

```
    Initiator                       Responder
   -----------                     -----------
    HDR*, HASH, N/D      -->
```

where N/D is either an ISAKMP notify payhload or an ISAKMP delete payload, and HASH is the prf output, using SKEYID_a as the key and the message ID from the ISAKMP header concatenated with the entire Informational payload (the Notify or Delete payload and any additional chained payloads) as the data. In other words, the hash for the above exchange is:

```
   HASH = prf(SKEYID_a, M-ID | N/D)
```

6.4.2 Acknowledged Informational

IKE exchanges are sent as ISAKMP messages whose delivery is unreliable. This fact has been taken into account in the design of Phase 1 and Phase 2 exchanges since retransmission timers are set to allow for packet loss. Due to the unreliable nature of the exchange described in 6.4.1 certain messages should not be sent that way. For instance, if a notification that an SA has been deleted is lost the sender may delete it but the intended recipient will have not way of knowing this fact.

The first payload of the acknowledged Informational exchange MUST be a hash payload and the second payload MUST be the nonce of the

sender. The Responder MUST NOT change the notification payloads which follow the Initiator's nonce.

The acknowledged Informational exchange is defined as:

```
    Initiator                       Responder
   -----------                     -----------
    HDR*, HASH(1), Ni, N/D  -->
                            <--     HDR*, HASH(2), Nr, N/D
```

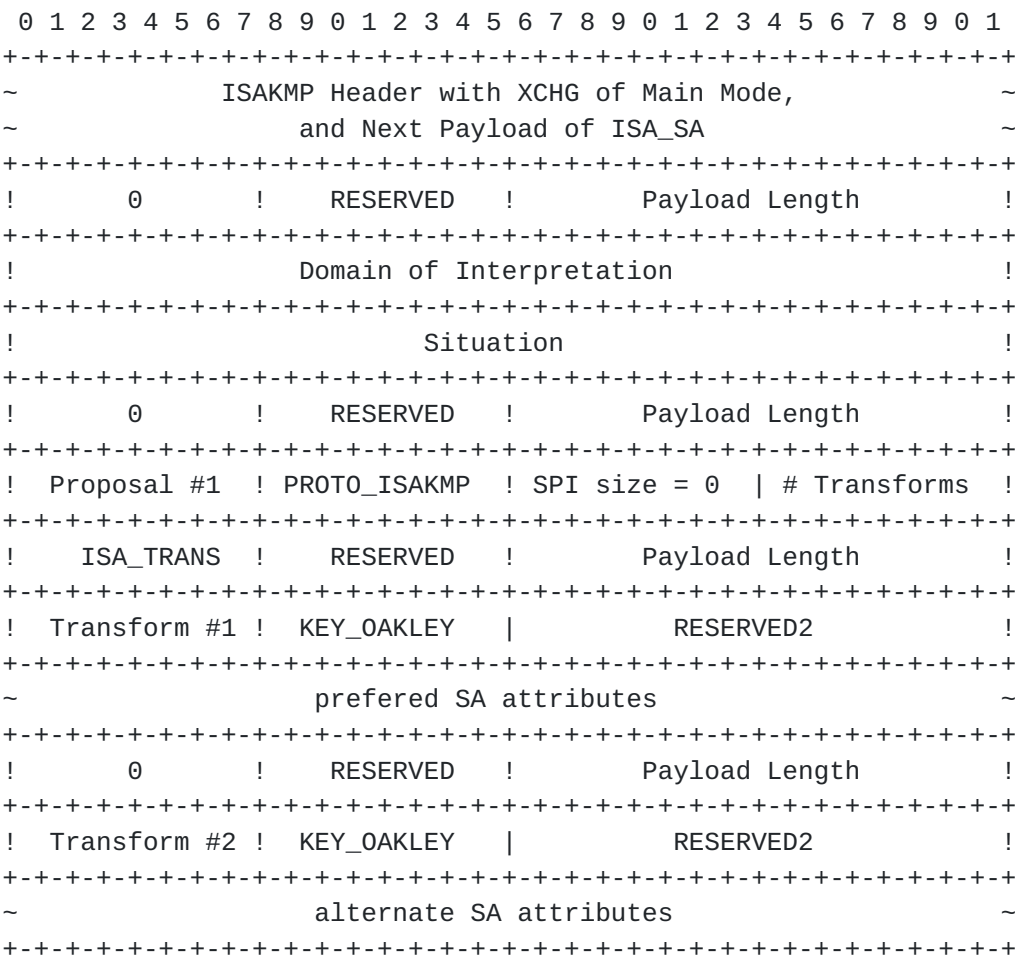
where HASH(1) is prf output using SKEYID_a from the Phase 1 SA identified by the cookies in the ISAKMP header as the key, and the unique, pseudo-random message ID for this exchange concatenated with the remaining payloads (in this case, a delete or notify and a nonce payload, but more payloads could be chained to this message) as the data, and HASH(2) is the prf output using SKEYID_a, from the Phase 1 SA identified by the cookies in the ISKAMP header, as the key and the nonce payload sent by the Initiator concatenated with the pseudo-random message ID for this exchange and the remaining payloads as the message to hash. The hashes for the above exchange would be:

```
HASH(1) = prf(SKEYID_a, M-ID | Ni | N/D)
HASH(2) = prf(SKEYID_a, Ni | M-ID | Nr | N/D)
```

This memo does not proscribe which messages should be sent with the Acknowledged or Unacknowledged Informational.

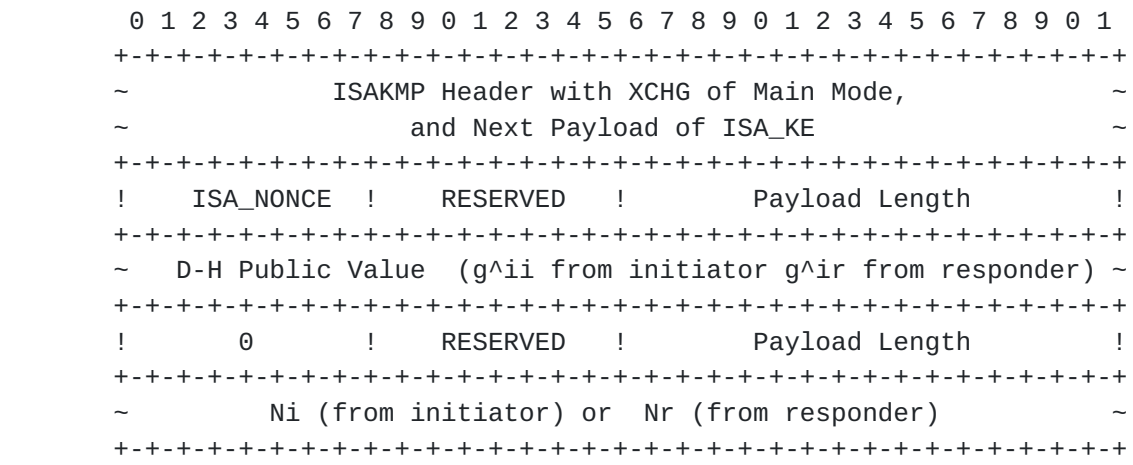
7 Payload Explosion of Sample Exchange

7.1 Phase 1 Using Main Mode

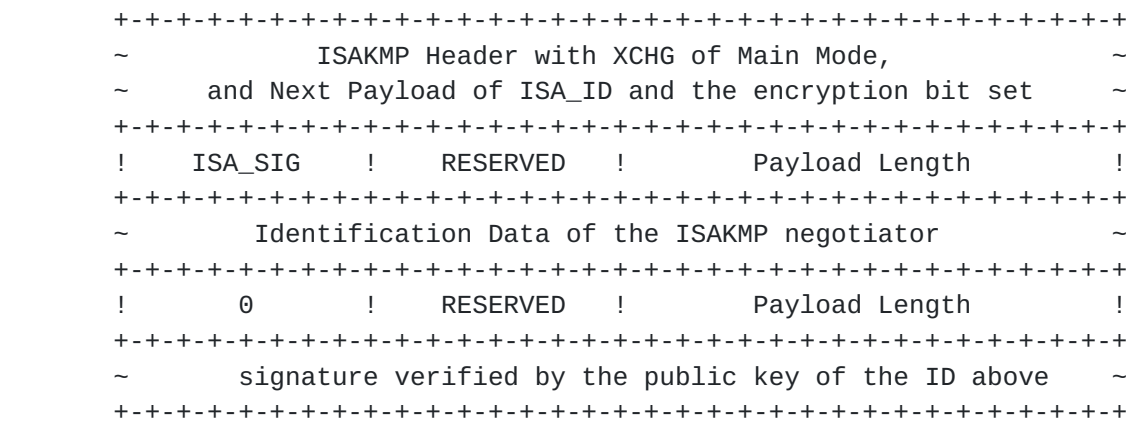


The responder replies in kind but selects, and returns, one transform proposal (the ISAKMP SA attributes).

The second exchange consists of the following payloads:



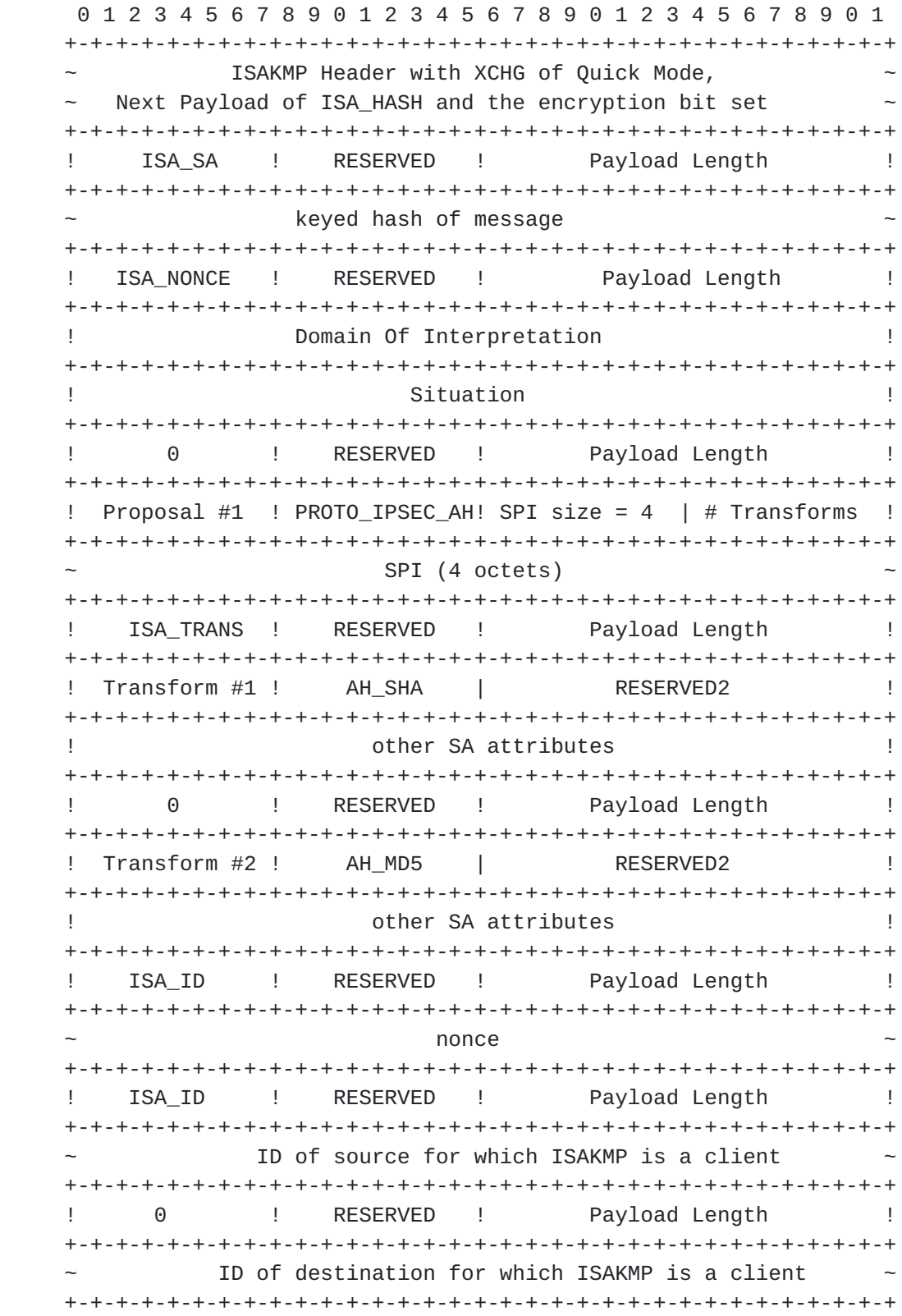
The shared keys, SKEYID_e and SKEYID_a, are now used to protect and authenticate all further communication. Note that both SKEYID_e and SKEYID_a are unauthenticated.



The key exchange is authenticated over a signed hash as described in section 5.1. Once the signature has been verified using the authentication algorithm negotiated as part of the ISAKMP SA, the shared keys, SKEYID_e and SKEYID_a can be marked as authenticated. (For brevity, certificate payloads were not exchanged).

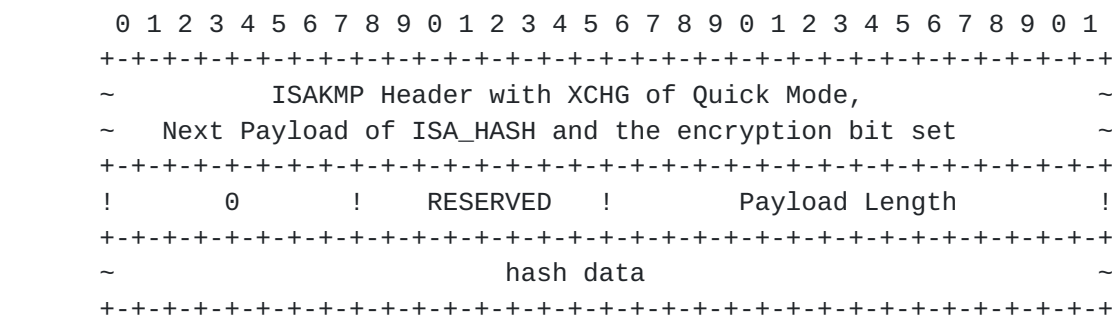
7.2 Phase 2 Using Quick Mode

The following payloads are exchanged in the first round of Quick Mode with ISAKMP SA negotiation. In this hypothetical exchange, the ISAKMP negotiators are proxies for other parties which have requested authentication.



where the contents of the hash are described in 5.5 above. The responder replies with a similar message which only contains one

transform-- the selected AH transform. Upon receipt, the initiator can provide the key engine with the negotiated security association and the keying material. As a check against replay attacks, the responder waits until receipt of the next message.



where the contents of the hash are described in 5.5 above.

8 Security Considerations

This entire memo discusses a hybrid protocol, combining parts of Oakley and parts of SKEME with ISAKMP, to negotiate, and derive keying material for, security associations in a secure and authenticated manner.

Confidentiality is assured by the use of a negotiated encryption algorithm. Authentication is assured by the use of a negotiated method: a digital signature algorithm; a public key algorithm which supports encryption; or, a pre-shared key. The confidentiality and authentication of this exchange is only as good as the attributes negotiated as part of the ISAKMP security association.

Repeated re-keying using Quick Mode without PFS can consume the entropy of the Diffie-Hellman shared secret. Implementors should take note of this fact and set a limit on Quick Mode Exchanges between exponentiations. This memo does not prescribe such a limit.

Perfect Forward Secrecy (PFS) of both keying material and identities is possible with this protocol. By using the PFS option during a Quick Mode, specifying a Diffie-Hellman group and passing public values in KE payloads, IKE peers can establish PFS of keys. The identities would be protected by SKEYID_e from the Phase 1 SA and would therefore not be protected by PFS. If PFS of both keying material and identities is desired, an IKE peer MUST establish only one non-ISAKMP security association (e.g. IPsec Security Association) per Phase 1 SA. PFS for keys and identities is accomplished by deleting the Phase 1 SA (and optionally issuing a DELETE message) upon establishment of the single Phase 2 SA. In this way the Phase one negotiation is uniquely tied to a single Phase two

negotiation, and is never used again.

The strength of a key derived from a Diffie-Hellman exchange using any of the groups defined here depends on the inherent strength of the group, the size of the exponent used, and the entropy provided by the random number generator used. Due to these inputs it is difficult to determine the strength of a key for any of the defined groups. The default Diffie-Hellman group (number two) when used with a strong random number generator and an exponent no less than 160 bits is sufficient to use for 3DES. Groups three through five provide greater security. Group one is for historic purposes only and does not provide sufficient strength to the required cipher (although it is sufficient for use with DES, which is also for historic use only). Implementations should make note of these conservative estimates when establishing policy and negotiating security parameters.

Note that these limitations are on the Diffie-Hellman groups themselves. There is nothing in IKE which prohibits using stronger groups nor is there anything which will dilute the strength obtained from stronger groups. In fact, the extensible framework of IKE encourages the definition of more groups; use of elliptical curve groups will greatly increase strength using much smaller numbers.

For situations where defined groups provide insufficient strength New Group Mode can be used to exchange a Diffie-Hellman group which provides the necessary strength. In is incumbent upon implementations to check the primality in groups being offered and independently arrive at strength estimates.

It is assumed that the Diffie-Hellman exponents in this exchange are erased from memory after use. In particular, these exponents must not be derived from long-lived secrets like the seed to a pseudo-random generator.

IKE exchanges maintain running initialization vectors (IV) where the last ciphertext block of the last message is the IV for the next message. To prevent retransmissions (or forged messages with valid cookies) from causing exchanges to get out of sync IKE implementations SHOULD NOT update their running IV until the decrypted message has passed a basic sanity check and has been determined to actually advance the IKE state machine-- i.e. it is not a retransmission.

While the last roundtrip of Main Mode (and optionally the last message of Aggressive Mode) is encrypted it is not, strictly speaking, authenticated. An active substitution attack on the ciphertext could result in payload corruption. If such an attack corrupts mandatory payloads it would be detected by an authentication

failure, but if it corrupts any optional payloads (e.g. notify payloads chained onto the last message of a Main Mode exchange) it might not be detectable.

The message ID used for all non-Phase 1 exchanges MUST be pseudo-randomly generated using a strong random number generator.

Optimal Asymmetric Encryption Padding [BR94] MUST be used with PKCS#1 to avoid the adaptive chosen ciphertext attack against RSA that is described in [Ble98]. See [PKCS1].

The acknowledged Informational exchange is open to replay attacks.

9 IANA Considerations

This document contains many "magic numbers" to be maintained by the IANA. This section explains the criteria to be used by the IANA to assign additional numbers in each of these lists.

9.1 Attribute Classes

Attributes negotiated in this protocol are identified by their class. Requests for assignment of new classes must be accompanied by a standards-track RFC which describes the use of this attribute.

9.2 Encryption Algorithm Class

Values of the Encryption Algorithm Class define an encryption algorithm to use when called for in this document. Requests for assignment of new encryption algorithm values must be accompanied by a reference to a standards-track or Informational RFC or a reference to published cryptographic literature which describes this algorithm.

9.3 Hash Algorithm

Values of the Hash Algorithm Class define a hash algorithm to use when called for in this document. Requests for assignment of new hash algorithm values must be accompanied by a reference to a standards-track or Informational RFC or a reference to published cryptographic literature which describes this algorithm. Due to the key derivation and key expansion uses of HMAC forms of hash algorithms in IKE, requests for assignment of new hash algorithm values must take into account the cryptographic properties-- e.g it's resistance to collision-- of the hash algorithm itself.

9.4 Group Description and Group Type

Values of the Group Description Class identify a group to use in a

Diffie-Hellman exchange. Values of the Group Type Class define the type of group. Requests for assignment of new groups must be accompanied by a reference to a standards-track or Informational RFC which describes this group. Requests for assignment of new group types must be accompanied by a reference to a standards-track or Informational RFC or by a reference to published cryptographic or mathmatical literature which describes the new type.

9.5 Life Type

Values of the Life Type Class define a type of lifetime to which the ISAKMP Security Association applies. Requests for assignment of new life types must be accompanied by a detailed description of the units of this type and its expiry.

10 Acknowledgements

This document is the result of close consultation with Hugo Krawczyk, Douglas Maughan, Hilarie Orman, Mark Schertler, Mark Schneider, and Jeff Turner. It relies on protocols which were written by them. Without their interest and dedication, this would not have been written.

Special thanks Rob Adams, Cheryl Madson, Derrell Piper, Harry Varnis, and Elfed Weaver for technical input, encouragement, and various sanity checks along the way. We would also like to thank Ben Rogers and D. Hugh Redelmeier for helping close up various holes in the text which were open to interpretation.

We would also like to thank the many members of the IPSec working group that contributed to the development of this protocol over the past two years, especially those of you who've implemented IKE. Thanks!

11 References

[CAST] Adams, C., "The CAST-128 Encryption Algorithm", RFC 2144, May 1997.

[BLOW] Schneier, B., "The Blowfish Encryption Algorithm", Dr. Dobb's Journal, v. 19, n. 4, April 1994.

[Bra96] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[Bra97] Bradner, S., "Key Words for use in RFCs to indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[Ble98] Bleichenbacher, D., "Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS#1", Advances in Cryptology Eurocrypt '98, Springer-Verlag, 1998.

[BR94] Bellare, M., and Rogaway P., "Optimal Asymmetric Encryption."
Advances in Cryptology Eurocrypt '94, Springer-Verlag, 1994.

[DES] ANSI X3.106, "American National Standard for Information Systems-Data Link Encryption", American National Standards Institute, 1983.

[DH] Diffie, W., and Hellman M., "New Directions in Cryptography", IEEE Transactions on Information Theory, V. IT-22, n. 6, June 1977.

[DSS] NIST, "Digital Signature Standard", FIPS 186, National Institute of Standards and Technology, U.S. Department of Commerce, May, 1994.

[IDEA] Lai, X., "On the Design and Security of Block Ciphers," ETH Series in Information Processing, v. 1, Konstanz: Hartung-Gorre Verlag, 1992

[KBC96] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[SKEME] Krawczyk, H., "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", from IEEE Proceedings of the 1996 Symposium on Network and Distributed Systems Security.

[MD5] Rivest, R., "The MD5 Message Digest Algorithm", RFC 1321, April 1992.

[MSST98] Maughhan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[Orm96] Orman, H., "The Oakley Key Determination Protocol", RFC 2412, November 1998.

[PKCS1] Kaliski, B., and J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2", September 1998.

[Pip98] Piper, D., "The Internet IP Security Domain Of Interpretation for ISAKMP", RFC 2407, November 1998.

[RC5] Rivest, R., "The RC5 Encryption Algorithm", Dr. Dobb's Journal, v. 20, n. 1, January 1995.

[RSA] Rivest, R., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, v. 21, n. 2, February 1978.

[Sch96] Schneier, B., "Applied Cryptography, Protocols, Algorithms, and Source Code in C", 2nd edition.

[SHA] NIST, "Secure Hash Standard", FIPS 180-1, National Institue of Standards and Technology, U.S. Department of Commerce, May 1994.

[TIGER] Anderson, R., and Biham, E., "Fast Software Encryption", Springer LNCS v. 1039, 1996.

Appendix A

Attribute Assigned Numbers

Attributes negotiated during phase one use the following definitions. Phase two attributes are defined in the applicable DOI specification (for example, IPsec attributes are defined in the IPsec DOI), with the exception of a group description when Quick Mode includes an ephemeral Diffie-Hellman exchange. Attribute types can be either Basic (B) or Variable-length (V). Encoding of these attributes is defined in [MSST98] as Type/Value (Basic) and Type/Length/Value (Variable).

Attributes described as basic MUST NOT be encoded as variable. Variable length attributes MAY be encoded as basic attributes if their value can fit into two octets. If this is the case, an attribute offered as variable (or basic) by the initiator of this protocol MAY be returned to the initiator as a basic (or variable).

Attribute Classes

class	value	type
Encryption Algorithm	1	B
Hash Algorithm	2	B
Authentication Method	3	B
Group Description	4	B
Group Type	5	B
Group Prime/Irreducible Polynomial	6	V
Group Generator One	7	V
Group Generator Two	8	V
Group Curve A	9	V
Group Curve B	10	V
Life Type	11	B
Life Duration	12	V
PRF	13	B
Key Length	14	B
Field Size	15	B
Group Order	16	V
Block Size	17	B

values 18-16383 are reserved to IANA. Values 16384-32767 are for private use among mutually consenting parties.

Class Values

- Encryption Algorithm		Defined In
DES-CBC	1	RFC 2405

IDEA-CBC	2
Blowfish-CBC	3
RC5-R16-B64-CBC	4
3DES-CBC	5
CAST-CBC	6

values 7-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties.

- Hash Algorithm

		Defined In
MD5	1	RFC 1321
SHA	2	FIPS 180-1
Tiger	3	See Reference [TIGER]

values 4-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties.

- Authentication Method

pre-shared key	1
DSS signatures	2
RSA signatures	3
Encryption with RSA	4
Revised encryption with RSA	5
Encryption with El-Gamal	6
Revised encryption with El-Gamal	7

values 8-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties.

- Group Description

768-bit MODP group (section 5.1)	1
1024-bit MODP group (section 5.2)	2
EC2N group on GP[2^155] (section 5.3)	3
EC2N group on GP[2^185] (section 5.4)	4
1536-bit MODP group (section 5.5)	5

values 6-32767 are reserved to IANA. Values 32768-65535 are for private use among mutually consenting parties.

- Group Type

MODP (modular exponentiation group)	1
ECP (elliptic curve group over GF[P])	2
EC2N (elliptic curve group over GF[2^N])	3

values 4-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties.

- Life Type

seconds	1
kilobytes	2

  values 3-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties. For a given "Life Type" the value of the "Life Duration" attribute defines the actual length of the SA life-- either a number of seconds, or a number of kbytes protected.

- PRF

  There are currently no pseudo-random functions defined.

  values 1-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties.

- Key Length

  When using an Encryption Algorithm that has a variable length key, this attribute specifies the key length in bits. (MUST use network byte order). This attribute MUST NOT be used when the specified Encryption Algorithm uses a fixed length key.

- Field Size

  The field size, in bits, of a Diffie-Hellman group.

- Group Order

  The group order of an elliptical curve group. Note the length of this attribute depends on the field size.

- Block Size

  The number of bits per block of a cipher with a variable block length.

Additional Exchanges Defined-- XCHG values

Quick Mode	32
New Group Mode	33
Acknowledged Informational	34

Appendix B

This appendix describes encryption details to be used ONLY when encrypting IKE messages. When a service (such as IPSec) utilizes IKE to generate keying material, all encryption algorithm specific details (such as key and IV generation, padding, etc...) MUST be defined by that service. IKE does not purport to produce keys that are suitable for every encryption algorithm. IKE produces the requested amount of keying material from which the service MUST generate a suitable key. Details, such as weak key checks, are the responsibility of the service.

Use of negotiated PRFs may require the prf output to be expanded due to the prf feedback mechanism employed by this document. For example, if the (ficticious) DOORAK-MAC requires 24 bytes of key but produces only 8 bytes of output, the output must be expanded three times before being used as the key for another instance of itself. The output of a prf is expanded by feeding back the results of the prf into itself to generate successive blocks. These blocks are concatenated until the requisite number of bytes has been acheived. For example, for pre-shared key authentication with DOORAK-MAC as the negotiated prf:

```
BLOCK1-8 = prf(pre-shared-key, Ni_b | Nr_b)
BLOCK9-16 = prf(pre-shared-key, BLOCK1-8 | Ni_b | Nr_b)
BLOCK17-24 = prf(pre-shared-key, BLOCK9-16 | Ni_b | Nr_b)
 and
SKEYID = BLOCK1-8 | BLOCK9-16 | BLOCK17-24
```

so therefore to derive SKEYID_d:

```
BLOCK1-8 = prf(SKEYID, g^ir | CKY-I | CKY-R | 0) BLOCK9-16 =
prf(SKEYID, BLOCK1-8 | g^ir | CKY-I | CKY-R | 0) BLOCK17-24 =
prf(SKEYID, BLOCK9-16 | g^ir | CKY-I | CKY-R | 0)
```

and

```
SKEYID_d = BLOCK1-8 | BLOCK9-16 | BLOCK17-24
```

Subsequent prf derivations are done similarly.

Encryption keys used to protect the Phase 1 SA are derived from SKEYID_e in an algorithm-specific manner. When SKEYID_e is not long enough to supply all the necessary keying material an algorithm requires, the key is derived from feeding the results of a pseudo-random function into itself, concatenating the results, and taking the highest necessary bits.

For example, if (ficticious) algorithm AKULA requires 320-bits of key (and has no weak key check) and the prf used to generate SKEYID_e only generates 120 bits of material, the key for AKULA, would be the first 320-bits of Ka, where:

```
Ka = K1 | K2 | K3
```

and

```
K1 = prf(SKEYID_e, 0)
K2 = prf(SKEYID_e, K1)
K3 = prf(SKEYID_e, K2)
```

where prf is the negotiated prf or the HMAC version of the negotiated hash function (if no prf was negotiated) and 0 is represented by a single octet. Each result of the prf provides 120 bits of material for a total of 360 bits. AKULA would use the first 320 bits of that 360 bit string.

The input to a CBC mode operation must be equivalent to the block size of the underlying cipher. To satisfy this requirment, IKE messages are padded up to the nearest block size using bytes containing 0x00. The message length in the ISAKMP header MUST include the length of this pad.

The key for 3DES-CBC is the first twenty-four (24) bytes of a key derived in the aforementioned pseudo-random function feedback method. 3DES-CBC is an encrypt-decrypt-encrypt operation using the first, middle, and last eight (8) bytes of the entire 3DES-CBC key. The IV is the first eight (8) bytes of the IV material derived in section 4.1 above.

The key for DES-CBC is derived from the first eight (8) non-weak and non-semi-weak (see Appendix A) bytes of SKEYID_e. The IV is the first 8 bytes of the IV material derived in section 4.1 above.

The key for IDEA-CBC is derived from the first sixteen (16) bytes of SKEYID_e. The IV is the first eight (8) bytes of the IV material derived in section 4.1 above.

The key for Blowfish-CBC is either the negotiated key size, or the first fifty-six (56) bytes of a key (if no key size is negotiated) derived in the aforementioned pseudo-random function feedback method. The IV is the first eight (8) bytes of the IV material derived in section 4.1 above.

The key for RC5-R16-B64-CBC is the negotiated key size, or the first sixteen (16) bytes of a key (if no key size is negotiated) derived

from the aforementioned pseudo-random function feedback method if necessary. The IV is the first eight (8) bytes of the IV material derived in section 4.1 above. The number of rounds MUST be 16 and the block size MUST be 64.

The key for CAST-CBC is either the negotiated key size, or the first sixteen (16) bytes of a key derived in the aforementioned pseudo-random function feedback method. The IV is the first eight (8) bytes of the IV material derived in section 4.1 above.

Support for algorithms other than 3DES-CBC is purely optional. Some optional algorithms may be subject to intellectual property claims.

Authors' Addresses

Dan Harkins
Network Alchemy
1615 Pacific ave
Santa Cruz, California, 95060
United States of America

Phone: +1 831 460 3800
EMail: dharkins@network-alchemy.com

Dave Carrel
Redback Networks
1389 Moffett Park Drive
Sunnyvale, CA, 94089-1134
United States of America

Phone: +1 408-548-3500
EMail: carrel@rback.net

Authors' Note

The authors encourage independent implementation, and interoperability testing, of this hybrid protocol.