The Oakley Key Determination Protocol

Hilarie Orman
University of Arizona[1]

TR 97 02

Abstract

This document describes a protocol, named OAKLEY, by which two authenticated parties can agree on secure and secret keying material. The basic mechanism is the Diffie-Hellman key exchange algorithm. The OAKLEY protocol supports Perfect Forward Secrecy, compatibility with the ISAKMP protocol for managing security associations, user-defined abstract group structures for use with the Diffie-Hellman algorithm, key updates, and incorporation of keys distributed via out-of-band mechanisms.

February 17, 1997

Department of Computer Science
The University of Arizona
Tucson, AZ 85721

[1] Authors' address: Department of Computer Science, University of Arizona, Tucson, AZ 85721. Email: {ho}@cs.arizona.edu.

IPSEC Working Group H. K. Orman
INTERNET-DRAFT Dept. of Computer Science, Univ. of Arizona
draft-ietf-ipsec-oakley-01.txt May 1996

The OAKLEY Key Determination Protocol
<draft-ietf-ipsec-oakley-01.txt>

This document describes a protocol, named OAKLEY, by which two authenticated parties can agree on secure and secret keying material. The basic mechanism is the Diffie-Hellman key exchange algorithm.

The OAKLEY protocol supports Perfect Forward Secrecy, compatibility with the ISAKMP protocol for managing security associations, user-defined abstract group structures for use with the Diffie-Hellman algorithm, key updates, and incorporation of keys distributed via out-of-band mechanisms.

Status of this Memo

Coast), or ftp.isi.edu (US West Coast).

Distribution of this memo is unlimited.

1. INTRODUCTION

Key establishment is the heart of data protection that relies on cryptography, and it is an essential component of the packet protection mechanisms described in [RFC1825, RFC1826, RFC1827], for example. A scalable and secure key distribution mechanism for the Internet is a necessity. The goal of this protocol is to provide that mechanism, coupled with a great deal of cryptographic strength.

The Diffie-Hellman key exchange algorithm provides such a mechanism. It allows two parties to agree on a shared value without requiring encryption. The shared value is immediately available for use in encrypting subsequent conversation, e.g. data transmission and/or authentication. The STS protocol [STS] provides a demonstration of how to embed the algorithm in a secure protocol, one that ensures that in addition to securely sharing a secret, the two parties can be sure of each other's identities, even when an active attacker exists.

Because OAKLEY is a generic key exchange protocol, and because the keys that it generates might be used for encrypting data with a long privacy lifetime, 20 years or more, it is important that the algorithms underlying the protocol be able to ensure the security of the keys for that period of time, based on the best prediction capabilities available for seeing into the mathematical future. The protocol therefore has two options for adding to the difficulties faced by an attacker who has a large amount of recorded key exchange traffic at his disposal (a passive attacker). These options are useful for deriving keys which will be used for encryption.

The OAKLEY protocol is related to STS, sharing the similarity of authenticating the Diffie-Hellman exponentials and using them for determining a shared key, and also of achieving Perfect Forward Secrecy for the shared key, but it differs from the STS protocol in several ways.

The first is the addition of a weak address identification mechanism ("cookies", described by Phil Karn [Photuris]) to help avoid denial of service attacks.

The second extension is to allow the two parties to select mutually agreeable supporting algorithms for the protocol: the encryption method, the key derivation method, and the authentication method.

Thirdly, the authentication does not depend on encryption using the Diffie-Hellman exponentials; instead, the authentication validates the binding of the exponentials to the identities of the parties.

The protocol does not require the two parties compute the shared exponentials prior to authentication.

This protocol adds additional security to the derivation of keys meant for use with encryption (as opposed to authentication) by including a dependence on an additional algorithm. The derivation of keys for encryption is made to depend not only on the Diffie-Hellman algorithm, but also on the cryptographic method used to securely authenticate the communicating parties to each other.

Finally, this protocol explicitly defines how the two parties can select the mathematical structures (group representation and operation) for performing the Diffie-Hellman algorithm; they can use standard groups or define their own. User-defined groups provide an additional degree of long-term security.

OAKLEY has several options for distributing keys. In addition to the classic Diffie-Hellman exchange, this protocol can be used to derive a new key from an existing key and to distribute an externally derived key by encrypting it.

The protocol allows two parties to use all or some of the anti-clogging and perfect forward secrecy features. It also permits the use of authentication based on symmetric encryption or non-encryption algorithms. This flexibility is included in order to allow the parties to use the features that are best suited to their security and performance requirements.

This document draws extensively in spirit and approach from the Photuris draft by Karn and Simpson [Photuris] (and from discussions with the authors), specifics of the ISAKMP draft by Schertler et al. [ISAKMP], and it was also influenced by papers by Paul van Oorschot and Hugo Krawcyzk.

2. The Protocol Outline

2.1 General Remarks

The OAKLEY protocol is used to establish a shared key with an assigned identifier and associated authenticated identities for the two parties. The name of the key can be used later to derive security associations for the RFC1826 and RFC1827 protocols (AH and ESP) or to achieve other network security goals.

Each key is associated with algorithms that are used for authentication, privacy, and one-way functions. These are ancillary algorithms for OAKLEY; their appearance in subsequent security association definitions derived with other protocols is neither required nor prohibited.

The specification of the details of how to apply an algorithm to data is called a transform. This document does not supply the transform definitions; they will be in separate RFC's.

The anti-clogging tokens, or "cookies", provide a weak form of source address identification for both parties; the cookie exchange can be completed before they perform the computationally expensive part of the protocol (large integer exponentiations).

It is important to note that OAKLEY uses the cookies for two purposes: anti-clogging and key naming. The two parties to the protocol each contribute one cookie at the initiation of key establishment; the pair of cookies becomes the key identifier (KEYID), a reusable name for the keying material. Because of this dual role, we will use the notation for the concatenation of the cookies ("COOKIE-I, COOKIE-R") interchangeably with the symbol "KEYID".

OAKLEY is designed to be a compatible component of the ISAKMP protocol [ISAKMP], which runs over the UDP protocol using a well-known port (see the RFC on port assignments, STD02-RFC-1700). The only technical requirement for the protocol environment is that the underlying protocol stack must be able to supply the Internet address of the remote party for each message. Thus, OAKLEY could, in theory, be used directly over the IP protocol or over UDP, if suitable protocol or port number assignments were available.

The machine running OAKLEY must provide a good random number generator, as described in [RFC1750], as the source of random numbers required in this protocol description. Any mention of a "nonce" implies that the nonce value is generated by such a generator. The same is true for "pseudorandom" values.

2.2 Notation

The section describes the notation used in this document for message sequences and content.

2.2.1 Message descriptions

The protocol exchanges below are written in an abbreviated notation that is intended to convey the essential elements of the exchange in a clear manner. A brief guide to the notation follows. The detailed formats and assigned values are given in the appendices.

In order to represent message exchanges succinctly, this document uses an abbreviated notation that describes each message in terms of its source and destination and relevant fields.

Arrows ("->") indicate whether the message sent from the initiator to the responder, or vice versa ("<-").

The fields in the message are named and comma separated. The protocol uses the convention that the first several fields constitute a fixed header format for all messages.

For example, consider a HYPOTHETICAL exchange of messages involving a fixed format message, the four fixed fields being two "cookies", the third field being a message type name, the fourth field being a multi-precision integer representing a power of a number:

```
     Initiator                                        Responder
         ->    Cookie-I, 0, OK_KEYX, g^x                     ->
```

```
          <-    Cookie-R, Cookie-I, OK_KEYX, g^y           <-
```

The notation describes a two message sequence. The initiator begins by sending a message with 4 fields to the responder; the first field has the unspecified value "Cookie-I", second field has the numeric value 0, the third field indicates the message type is OK_KEYX, the fourth value is an abstract group element g to the x'th power.

The second line indicates that the responder replies with value "Cookie-R" in the first field, a copy of the "Cookie-I" value in the second field, message type OK_KEYX, and the number g raised to the y'th power.

The value OK_KEYX is in capitals to indicate that it is a unique constant (constants are defined the appendices).

2.2.2 Guide to symbols

Cookie-I and Cookie-R (or CKY-I and CKY-R) are 64-bit pseudo-random numbers. The generation method must ensure with high probability that the numbers are unique over some previous time period, such as one hour.

KEYID is the concatenation of the initiator and responder cookies and the domain of interpretation; it is the name of keying material.

sKEYID is used to denote the keying material named by the KEYID. It is never transmitted, but it is used in various calculations performed by the two parties.

OK_KEYX, OK_NEWGRP, and OK_SET_DEF are distinct message types.

IDP is a bit indicating whether or not material after the encryption boundary (see appendix D), is encrypted.

g^x and g^y are encodings of group elements, where g is a special group element indicated in the group description (see Appendix Group Descriptors) and g^x indicates that element raised to the x'th power. The type of the encoding is either a variable precision integer or a pair of such integers, as indicated in the group operation in the group description. Note that we will write g^xy as a short-hand for g^(xy). See Appendix J for references that describe implementing large integer computations and the relationship between various group definitions and basic arithmetic operations.

EHAO is a list of encryption/hash/authentication choices. Each item is a of pair values: a class name and an algorithm name.

EHAS is a set of three items selected from the EHAO list, one from each of the classes for encryption, hash, authentication.

GRP is a name (32-bit value) for the group and its relevant parameters: the size of the integers, the arithmetic operation, and the generator element. There are a few pre-defined GRP's (for 768 bit modular exponentiation groups, 1024 bit modexp, 2048 bit modexp, 155-bit elliptic curve, see Appendix H), but participants can share other group descriptions in a later protocol stage (see the section NEW GROUP).

The symbol vertical bar "|" is used to denote concatenation of bit strings.

Ni and Nr are nonces selected by the initiator and responder, respectively.

ID(I) and ID(R) are the identities to be used in authenticating the initiator and responder respectively.

E{x}Ki indicates the encryption of x using the public key of the initiator. Encryption is done using the algorithm associated with the authentication method; usually this will be RSA.

S{x}Ki indicates the signature over x using the private key (signing key) of the initiator. Usually this will be RSA or DSS.

prf(a, b) denotes the result of applying pseudo-random function "a" to data "b".

prf(0, b) denotes the application of a one-way function to data "b". The similarity with the previous notation is deliberate and indicates that a single algorithm, e.g. MD5, might will used for both purposes. In the first case a "keyed" MD5 transform would be used with key "a"; in the second case the transform would have the fixed key value zero, resulting in a one-way function.

2.3 The Key Exchange Message Overview

The goal of key exchange processing of the secure establishment of common keying information state in the two parties. This state information is a key name, secret keying material, the identification of the two parties, and three algorithms for use during authentication: encryption (for privacy of the identities of the two parties), hashing (a pseudorandom function for protecting the integrity of the messages and for authentication), and authentication (the algorithm on which the authentication is based). The encodings and meanings for these choices are presented in Appendix B.

The main mode exchange has five optional features: stateless cookie exchange, perfect forward secrecy for the keying material, secrecy for the identities, perfect forward secrecy for identity secrecy, use of signatures (for non-repudiation). The two parties can use all or none of these features.

The general outline of processing is that the Initiator of the exchange begins by specifying as much information as he wishes in his first message. The Responder replies, supplying as much information as he wishes. The two sides exchange messages, supplying more information each time, until their requirements are satisfied.

The choice of how much information to include in each message depends on which options are desirable. For example, if stateless cookies are not a requirement, and identity secrecy and perfect forward secrecy for the keying material are not requirements, and if non-repudiatable signatures are acceptable, then the exchange can be completed in three messages.

Additional features may increase the number of roundtrips needed for the keying material determination.

ISAKMP provides fields for specifying the security association parameters for use with the AH and ESP protocols. These security association payload types are specified in the ISAKMP draft; the payload types can be protected with OAKLEY keying material and algorithms, but this document does not discuss their use.

2.3.1 The Essential Key Exchange Message Fields

There are 12 fields in an OAKLEY key exchange message. Not all the fields are relevant in every message; if a field is not relevant it can have a null value or not be present (no payload).

```
CK-I             originator cookie.
CK-R             responder cookie.
MSGTYPE          for key exchange, will be ISA_KE&AUTH_REQ or ISA_KE&AUTH_REP;
                 for new group definitions, will be ISA_NEW_GROUP_REQ
                 or ISA_NEW_GROUP_REP
GRP              the name of the Diffie-Hellman group used for the exchange
g^x (or g^y)     variable length integer representing a power of
                 group generator
EHAO or EHAS     encryption, hash, authentication functions, offered
                 and selected
IDP              an indicator as to whether or not encryption with
                 g^xy follows (perfect forward secrecy for ID's)
ID(I)            the identity for the Initiator
ID(R)            the identity for the Responder
Ni               nonce supplied by the Initiator
Nr               nonce supplied by the Responder
```

The construction of the cookies is implementation dependent. Phil Karn has recommended making them the result of a one-way function applied to a secret value (changed periodically), the local and remote IP address, and the local and remote UDP port. In this way, the cookies remain stateless and expire periodically. Note that with OAKLEY, this would cause the KEYID's derived from the secret value to also expire, necessitating the removal of any state information associated with it.

The encryption functions used with OAKLEY must be cryptographic transforms which guarantee privacy and integrity for the message data. Merely using DES in CBC mode is not permissible. The MANDATORY and OPTIONAL transforms will include any that satisfy this criteria and are defined for use with RFC1827 (ESP).

The one-way (hash) functions used with OAKLEY must be cryptographic transforms which can be used as either keyed hash (pseudo-random) or non-keyed transforms. The MANDATORY and OPTIONAL transforms will include any that are defined for use with RFC1826 (AH).

Where nonces are indicated, they will be variable precision integers with an entropy value that matches the "strength" attribute of the GRP used with the exchange. If no GRP is indicated, the nonces must be at least 90 bits long. The pseudo-random generator for the nonce material should start with initial data that has at least 90 bits of entropy; see RFC1750.

2.3.2 Mapping to ISAKMP Message Structures

All the OAKLEY message fields correspond to ISAKMP message payloads or payload components. The relevant payload fields are the SA payload, the AUTH payload, the Certificate Payload, the Key Exchange Payload.

Some of the ISAKMP header and payload fields will have constant values when used with OAKLEY:

DOI, the Domain of Interpretation, will have the value INTERNET. In this document, the DOI will not be mentioned; it is assumed that the software implementing OAKLEY will always be in the IPv4 or IPv6 DOI.

Unless otherwise noted, the Key Exchange Identifier is Oakley Main Mode.

In the SA Payload, the Situation is ISAKMPID

In the following we indicate where each OAKLEY field appears in the ISAKMP message structure.

CK-I	ISAKMP header
CK-R	ISAKMP header
MSGTYPE	Message Type in ISAKMP header
GRP	In ISAKMP header (replaces SPI's)
g^x (or g^y)	Key Exchange Payload, encoded as a variable precision integer
EHAO and EHAS	SA payload, Proposal section
IDP	Exchange field in the AUTH Payload header
ID(I)	AUTH payload, Identity field
ID(R)	AUTH payload, Identity field
Ni	AUTH payload, Nonce Field

```
Nr                 AUTH payload, Nonce Field
S{...}Kx           AUTH payload, Data Field
prf{K,...}         AUTH payload, Data Field
```

2.4 The Key Exchange Protocol

The exact number and content of messages exchanged during an OAKLEY key exchange depends on which options the Initiator and Responder want to use. A key exchange can be completed with three or more messages, depending on those options.

The three components of the key determination protocol are the
1. cookie exchange (optionally stateless)
2. Diffie-Hellman half-key exchange (optional, but essential for perfect forward secrecy)
3. authentication (options: privacy for ID's, privacy for ID's with PFS, non-repudiatable)

The initiator can supply as little information as a bare exchange exchange request, carrying no additional information. On the other hand the initiator can begin by supplying all of the information necessary for the responder to authenticate the request and complete the key determination quickly, if the responder chooses to accept this method. If not, the responder can reply with a minimal amount of information (at the minimum, a cookie).

The Initiator is responsible for retransmitting messages if the protocol does not terminate in a timely fashion. The Responder must therefore avoid discarding reply information until it is acknowledged by Initiator in the course of continuing the protocol.

The remainder of this section contains examples demonstrating how to use OAKLEY options.

2.4.1 An Aggressive Example

The following example indicates how two parties can complete a key exchange in two messages. The identities are not secret, the derived keying material is protected by PFS.

By using digital signatures, the two parties will have a proof of communication that can be recorded and presented later to a third party.

The keying material implied by the group exponentials is not needed for completing the exchange. If it is desirable to defer the computation, the implementation can save the "x" and "g^y" values and mark the keying material as "uncomputed". It can be computed from this information later.

```
   Initiator                                                    Responder
   ---------                                                    ---------
     -> CKY-I, 0,     OK_KEYX, GRP, g^x, EHAO, NIDP,               ->
        ID(I), ID(R), Ni,
        S{ID(I), ID(R), Ni, 0, GRP, g^x, EHAO}Ki
    <-  CKY-R, CKY-I, OK_KEYX, GRP, g^y, EHAS, NIDP,
```

```
      ID(R), ID(I), Nr, Ni
      S{ID(R), ID(I), Nr, Ni, GRP, g^y, EHAS}Kr               <-
   -> CKY-I, CKY-R, OK_KEYX, 0, 0, 0, NIDP,                    ->
      Ni, Nr, S{ID(I), ID(R), Ni, Nr}Ki
```

NB "NIDP" means that the PFS option for hiding identities is not used.
i.e., the identities are not encrypted using g^xy

The result of this exchange is a key with KEYID = CKY-I|CKY-R and value

sKEYID = prf(Ni | Nr, g^xy | CKY-I | CKY-R).

The processing outline for this exchange is as follows:

Initiation
The initiator generates a unique cookie and associates it with the expected IP address of the responder, and its chosen state information: GRP, g^x, EHAO list. The first authentication choice in the EHAO list is an algorithm that supports digital signatures, and this is used to sign the ID's and the nonce and group id. The initiator further

notes that the key is in the initial state of "unauthenticated", and

sets a timer for possible retransmission and/or termination of the request.

When responder receives the message, he may choose to ignore all the information and treat it as merely a request for a cookie, creating no state. If CKY-I is not already in use by the source address in the IP header, the responder generates a unique cookie, CKY-R. The next steps depend on the responders preferences. The minimal required response is to reply with the first cookie field set to zero and CKY-R in the second field. For this example we will assume that responder is more aggressive and accepts the following:
group GRP,
first authentication choice (which must be a digital signature),
lack of perfect forward secrecy for protecting the identities,
identity ID(I),
identity ID(R)

The responder must validate the signature over the signed portion of the message, and associate the pair (CKY-I, CKY-R) with the following state information:

the network address of the message

key state of "unauthenticated"

the first algorithm from the authentication offer

group GRP and a g^y value in group GRP

the nonce Ni and a pseudorandomly selected value Nr

a timer for possible destruction of the state.

The responder then signs the state information with the public key of ID(R) and sends it to the initiator.

In this example, to expedite the protocol, the responder implicitly accepts the first algorithm in the Authentication class of the EHAO list. This because he cannot validate the initiator signature without accepting the algorithm for doing the signature. The responder's EHAS list will also reflect his acceptance.

The initiator receives the reply message and
validates that CKY-I is a valid association for the network address of the incoming message,

adds the CKY-R value to the state for the pair (CKY-I, network address), and associates all state information with the pair (CKY-I, CKY-R),

adds g^y to its state information,

pseudorandomly chooses an exponent x and computes the corresponding g^x value,

saves the EHA selections in the state,

optionally computes (g^x)^y (= g^xy) at this point, and

sends the reply message, signed with the public key of ID(I).

marks the KEYID (CKY-I|CKY-R) as authenticated.

When the responder receives this message, it marks the key as being in the authenticated state. If it has not already done so, it should compute g^xy and associate it with the KEYID.

Note that although PFS for identity protection is not used, PFS for the derived keying material is still present because the Diffie-Hellman half-keys g^x and g^y are exchanged.

2.4.1.1 Signature via Pseudo-Random Functions

The aggressive example is written to suggest that public key technology is used for the signatures. However, a pseudorandom function can be used, if the parties have previously agreed to such a scheme and have a shared key.

If the first proposal in the EHAO list is an "existing key" method, then the KEYID named in that proposal will supply the keying material for the "signature" which is computed using the "H" algorithm associated with the KEYID.

Suppose the first proposal in EHAO is

```
          EXISTING-KEY, 32
```

and the "H" algorithm for KEYID 32 is MD5-HMAC, by prior negotiation. The keying material is some string of bits, call it sK32. Then in the first message in the aggressive exchange, where the signature

```
        S{ID(I), ID(R), Ni, 0, GRP, g^x, EHAO}Ki
```

is indicated, the signature computation would be performed by

```
    MD5-HMAC_func(KEY=sK32, DATA = ID(I) | ID(R) | Ni | 0 | GRP | g^x
   | EHAO)
```

(The exact definition of the algorithm corresponding to "MD5-HMAC-func" will appear in the RFC defining that transform).

The result of this computation appears in the Authentication payload.

2.4.2 An Aggressive Example With Hidden Identities

The following example indicates how two parties can complete a key exchange without using digital signatures. Public key cryptography hides the identities during authentication. The group exponentials are exchanged and authenticated, but the implied keying material (g^xy) is not needed during the exchange.

This exchange has an important difference from the previous signature scheme --- in the first message, an identity for the responder is indicated as cleartext: ID(R'). However, the identity hidden with the public key cryptography is different: ID(R). This happens because the Initiator must somehow tell the Responder which public/private key pair to use for the decryption, but at the same time, the identity is hidden by encryption with that public key.

The Initiator might elect to forgo secrecy of the Responder identity, but this is undesirable. Instead, if there is a well-known identity for the Responder node, the public key for that identity can be used to encrypt the actual Responder identity.

```
Initiator                                                     Responder
---------                                                     ---------
  -> CKY-I, 0,     OK_KEYX, GRP, g^x, EHAO, NIDP,                 ->
     ID(R'), E{ID(I), ID(R), E{Ni}Kr}Kr',
 <-  CKY-R, CKY-I, OK_KEYX, GRP, g^y, EHAS, NIDP,
     E{ID(R), ID(I), Nr}Ki,
     prf(Kir, ID(R) | ID(I) | Nr | Ni | GRP | g^y | g^x)        <-
  -> CKY-I, CKY-R, OK_KEYX, GRP, 0, 0, NIDP,
     prf(Kir, ID(I)| ID(R) | Ni | Nr | GRP | g^x | g^y)           ->
```

Kir = prf(0, Ni | Nr)

NB "NIDP" means that the PFS option for hiding identities is not used.

NB The ID(R') value is included in the Authentication payload as described in Appendix B.

The result of this exchange is a key with KEYID = CKY-I|CKY-R and value sKEYID = prf(Ni | Nr, g^x | g^y | CKY-I | CKY-R).

The processing outline for this exchange is as follows:

Initiation

The initiator generates a unique cookie and associates it with the expected IP address of the responder, and its chosen state information: GRP, g^x, EHAO list. The first authentication choice in the EHAO list is an algorithm that supports public key encryption. The initiator also names the two identities to be used for the connection and enters these into the state. A well-known identity for the responder machine is also chosen, and the public key for this identity is used to encrypt the nonce Ni and the two connection identities. The initiator further

notes that the key is in the initial state of "unauthenticated", and

sets a timer for possible retransmission and/or termination of the request.

When responder receives the message, he may choose to ignore all the information and treat it as merely a request for a cookie, creating no state.

If CKY-I is not already in use by the source address in the IP header, the responder generates a unique cookie, CKY-R. As before, the next steps depend on the responders preferences. The minimal required response is a message with the first cookie field set to zero and CKY-R in the second field. For this example we will assume that responder is more aggressive and accepts the following:

group GRP,
first authentication choice (which must be a public key encryption algorithm),
lack of perfect forward secrecy for protecting the identities,
identity ID(I),
identity ID(R)

The responder now associates the pair (CKY-I, CKY-R) with the following state information:

the network address of the message

key state of "unauthenticated"

the first algorithm from each class in the EHAO (encryption-hash-authentication algorithm offers) list

group GRP and a g^y value in group GRP

the nonce Ni and a pseudorandomly selected value Nr a timer for possible destruction of the state.

The responder then encrypts the state information with the public key of ID(I) and sends it to the initiator.

The initiator receives the reply message and

validates that CKY-I is a valid association for the network address of the incoming message,

adds the CKY-R value to the state for the pair (CKY-I, network address), and associates all state information with the pair (CKY-I, CKY-R),

adds g^y to its state information,

chooses an exponent x and computes the corresponding g^x value,

saves the EHA selections in the state,

optionally computes (g^x)^y (= g^xy) at this point, and

sends the reply message, encrypted with the public key of ID(R).

marks the KEYID (CKY-I|CKY-R) as authenticated.

When the responder receives this message, it marks the key as being in the authenticated state. If it has not already done so, it should compute g^xy and associate it with the KEYID.

The secret keying material sKEYID = prf(Ni | Nr, g^xy | CKY-I | CKY-R)

Note that although PFS for identity protection is not used, PFS for the derived keying material is still present because the Diffie-Hellman half-keys g^x and g^y are exchanged.

2.4.3 An Aggressive Example With Private Identities and Without Diffie-Hellman

Considerable computational expense can be avoided if perfect forward secrecy is not a requirement for the session key derivation. The two parties can exchange nonces and secret key parts to achieve the authentication and derive keying material. The long-term privacy of data protected with derived keying material is dependent on the private keys of each of the parties.

In this exchange, the GRP has the value 0 and the field for the group exponential is used to hold a nonce value instead.

As in the previous section, the first proposed algorithm must be a public key encryption system; by responding with a cookie and a non-zero exponential field, the Responder implicitly accepts the first proposal and the lack of perfect forward secrecy for the identities and derived keying material.

```
Initiator                                                    Responder
---------                                                    ---------
  -> CKY-I, 0,     OK_KEYX, 0, 0, EHAO, NIDP,                   ->
     ID(R'), E{ID(I), ID(R), sKi}Kr',
 <-  CKY-R, CKY-I, OK_KEYX, 0, 0, EHAS, NIDP,
     E{ID(R), ID(I), sKr}Ki,
     prf(Kir, ID(R), ID(I), Nr, Ni)                            <-
  -> CKY-I, CKY-R, OK_KEYX, EHAS, NIDP,
     prf(Kir, ID(I), ID(R), Ni, Nr)                             ->
```

Kir = prf(0, sKi | sKr)

NB "NIDP" means that the PFS option for hiding identities is not used.

The result of this exchange is a key with KEYID = CKY-I|CKY-R and value sKEYID = prf(Kir, CKY-I | CKY-R).

2.4.4 A Conservative Example

In this example the two parties are less aggressive; they use the cookie exchange to delay creation of state, and they use perfect forward secrecy to protect the identities.

The responder considers the ability of the initiator to repeat CKY-R as weak evidence that the message originates from a "live" correspondent on the network and the correspondent is associated with the initiator's network address. The initiator makes similar assumptions when CKY-I is repeated to the initiator.

All messages must have either have valid cookies or at least one zero cookie. If both cookies are zero, this indicates a request for a cookie; if only the initiator cookie is zero, it is a response to a cookie request.

Information in messages violating the cookie rules cannot be used for any OAKLEY operations.

Note that the Initiator must and Responder must agree on one set of EHA algorithms; there is not one set for the Responder and one for the Initiator. The Initiator must include at least MD5 and DES in the initial offer.

Fields not indicated have null values.

```
Initiator                                                  Responder
---------                                                  ---------
  ->     0, 0, OK_KEYX                                        ->
 <-      0, CKY-R, OK_KEYX                                   <-
  ->     CKY-I, CKY-R, OK_KEYX, GRP, g^x, EHAO                ->
```

```
 <-      CKY-R, CKY-I, OK_KEYX, GRP, g^y, EHAS                 <-
  ->     CKY-I, CKY-R, OK_KEYX, GRP, 0  , 0, IDP*,
         ID(I), ID(R), E{Ni}Kr,                                 ->
 <-      CKY-R, CKY-I, OK_KEYX, GRP, 0  , 0, IDP,              <-
         E{Nr, Ni}Ki, ID(R), ID(I),
         prf(Nr | Ni, GRP, g^xy, ID(R), ID(I))
  ->     CKY-I, CKY-R, OK_KEYX, GRP, 0  , 0, IDP,
         prf(Ni | Nr, GRP, g^xy, ID(I), ID(R))                  ->
```

* when IDP is in effect, authentication payloads are encrypted with the selected encryption algorithm using the key prf(0, g^xy. This is in addition to and after any public key encryption. See Appendix B.

The first exchange allows the Responder to use stateless cookies; if the responder generates cookies in a manner that allows him to validate them without saving them, as in Photuris, then this is possible. Even if the Initiator includes a cookie in his initial request, the responder can still use stateless cookies by merely omitting the CKY-I from his reply and by declining to record the Initiator cookie until it appears in a later message.

After the exchange is complete, both parties compute the shared key material sKEYID as
prf(Ni | Nr, g^xy | CKY-I | CKY-R)
where "prf" is the pseudo-random function in class "hash" selected in the EHA list.

As with the cookies, each party considers the ability of the remote side to repeat the Ni or Nr value as a proof that Ka, the public key of party a, speaks for the remote party and establishes its identity.

In analyzing this exchange, it is important to note that although the IDP option ensures that the identities are protected with an ephemeral key g^xy, the authentication itself does not depend on g^xy. It is essential that the authentication steps validate the g^x and g^y values, and it is thus imperative that the authentication not involve a circular dependency on them. A third party could intervene with a "man-in-middle" scheme to convince the initiator and responder to use different g^xy values; although such an attack might result in revealing the identities to the eavesdropper, the authentication would fail.

2.4.5 Extra Strength for Protection of Encryption Keys

The nonces Ni and Nr are used to provide an extra dimension of secrecy in deriving session keys. This makes the secrecy of the key depend on two different problems: the discrete logarithm problem in the group G, and the problem of breaking the nonce encryption scheme. If RSA encryption is used, then this second problem is roughly equivalent to factoring the RSA public keys of both the initiator and responder.

For authentication, the key type, the validation method, and the certification requirement must be indicated.

2.5 Identity and Authentication

2.5.1 Identity

In OAKLEY exchanges the Initiator offers Initiator and Responder ID's -- the former is the claimed identity for the Initiator, and the latter is the requested ID for the Responder.

If neither ID is specified, the ID's are taken from the IP header source and destination addresses.

If the Initiator doesn't supply a responder ID, the Responder can reply by naming any identity that the local policy allows. The Initiator can refuse acceptance by terminating the exchange.

The Responder can also reply with a different ID than the Initiator suggested; the Initiator can accept this implicitly by continuing the exchange or deny by terminating.

2.5.2 Authentication

The authentication of principals to one another is at the heart of any key exchange scheme. The Internet community must decide on a scalable standard for solving this problem, and OAKLEY must make use of that standard. At the time of this writing, there is no such standard, though several are emerging. This document attempts to describe how a handful of standards could be incorporated into OAKLEY, without attempting to pick and choose among them.

The following methods can appear in OAKLEY offers:

a. Pre-shared Keys
   When two parties have arranged for a trusted method of distributing secret keys for their mutual authentication, they can be used for authentication. This has obvious scaling problems for large systems, but it is an acceptable interim solution for some situations. Support for pre-shared keys is REQUIRED.

   The encryption, hash, and authentication algorithm for use with a pre-shared key must be part of the state information distributed with the key itself.

   The pre-shared keys have a KEYID and keying material sKEYID; the KEYID is used in a pre-shared key authentication option offer.

There can be more than one pre-shared key offer in a list.

Because the KEYID persists over different invocations of OAKLEY (after a crash, etc.), it must occupy a reserved part of the KEYID space for the two parties. A few bits can be set aside in each party's "cookie space" to accommodate this.

There is no certification authority for pre-shared keys. When a pre-shared key is used to generate an authentication payload, the certification authority is "None", the Authentication Type is "Preshared", and the payload contains

the KEYID, encoded as two 64-bit quantities, and

the result of applying the pseudorandom hash function to the message body with the sKEYID forming the key for the function

See Appendix B for details of formats for the Authentication Payload.

b. DNS public keys

Security extensions to the DNS protocol [DNSSEC] provide a convenient way to access public key information, especially for public keys associated with hosts. RSA keys are a requirement for secure DNS implementations; extensions to allow optional DSS keys are a near-term possibility.

DNS KEY records have associated SIG records that are signed by a zone authority, and a hierarchy of signatures back to the root server establishes a foundation for trust. The SIG records indicate the algorithm used for forming the signature.

OAKLEY implementations MUST support the use of DNS KEY and SIG records for authenticating with respect to IPv4 and IPv6 addresses and fully qualified domain names. However, implementations are not required to support any particular algorithm (RSA, DSS, etc.).

c. RSA public keys w/o certification authority signature

PGP [Zimmerman] uses public keys with an informal method for establishing trust. The format of PGP public keys and naming methods will be described in a separate RFC. The RSA algorithm can be used with PGP keys for either signing or encryption; the authentication option should indicate either RSA-SIG or RSA-ENC, respectively. Support for this is OPTIONAL.

d.1 RSA public keys w/ certificates

There are various formats and naming conventions for public keys that are signed by one or more certification authorities. The Public Key Interchange Protocol discusses X.509 encodings and validation. Support for this is OPTIONAL.

d.2 DSS keys w/ certificates

Encoding for the Digital Signature Standard with X.509 is described in in draft-ietf-ipsec-dss-cert-00.txt. Support for this is OPTIONAL; an ISAKMP Authentication Type will be assigned.

2.5.3 Validating Authentication Keys

The combination of the Authentication algorithm, the Authentication Authority, the Authentication Type, and a key (usually public) define how to validate the messages with respect to the claimed identity. The key information will be available either from a pre-shared key, or from some kind of certification authority.

Generally the certification authority produces a certificate binding the entity name to a public key. OAKLEY implementations must be prepared to fetch anad validate certificates before using the public key for OAKLEY authentication purposes.

The ISAKMP Authentication Payload defines the Authentication Authority field for specifying the authority that must be apparent in the trust hierarchy for authentication.

Once an appropriate certificate is obtained (see 2.4.3), the validation method will depend on the Authentication Type; if it is PGP then the PGP signature validation routines can be called to satisfy the local web-of-trust predicates; if it is RSA with X.509 certificates, the certificate must be examined to see if the certification authority signature can be validated, and if the hierarchy is recognized by the local policy.

2.5.4 Fetching Identity Objects

In addition to interpreting the certificate or other data structure that contains an identity, users of OAKLEY must face the task of retrieving certificates that bind a public key to an identifier and also retrieving auxiliary certificates for certifying authorities or co-signers (as in the PGP web of trust).

The ISAKMP Credentials Payload can be used to attach useful certificates to OAKLEY messages. The Credentials Payload is defined in Appendix B.

Support for accessing and revoking public key certificates via the Secure DNS protocol [SECDNS] is MANDATORY for OAKLEY implementations. Other retrieval methods can be used when the AUTH class indicates a preference.

The Public Key Interchange Protocol discusses a full protocol that might be used with X.509 encoded certificates.

2.6 Interface to Cryptographic Transforms

The keying material computed by the key exchange should have at least 90 bits of entropy, which means that it must be at least 90 bits in length. This may be more or less than is required for keying the encryption and/or pseudorandom function transforms.

The transforms used with OAKLEY should have auxiliary algorithms which take a variable precision integer and turn it into keying material of the appropriate length. For example, a DES algorithm could take the low order 56 bits, a triple DES algorithm might use the following:

```
           K1 = low 56 bits of md5(0|sKEYID)
           K2 = low 56 bits of md5(1|sKEYID)
           K3 = low 56 bits of md5(2|sKEYID)
```

The transforms will be called with the keying material encoded as a variable precision integer, the length of the data, and the block of memory with the data. Conversion of the keying material to a transform key is the responsibility of the transform.

2.7 Retransmission, Timeouts, and Error Messages

If a response from the Responder is not elicited in an appropriate amount of time, the message should be retransmitted by the Initiator. These retransmissions must be handled gracefully by both parties; the Responder must retain information for retransmitting until the Initiator moves to the next message in the protocol or completes the exchange.

Informational error messages present a problem because they cannot be authenticated using only the information present in an incomplete exchange; for this reason, the parties may wish to establish a default key for OAKLEY error messages. A possible method for establishing such a key is described in Appendix B, under the use of ISA_INIT message types.

In the following the message type is OAKLEY Error, the KEYID supplies the H algorithm and key for authenticating the message contents; this value is carried in the Sig/Prf payload.

The Error payload contains the error code and the contents of the rejected message.

```
                          1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     !                                                               !
     ~                       Initiator-Cookie                        ~
  /  !                                                               !
KEYID +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  \  !                                                               !
```

```
~                        Responder-Cookie                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                  Domain of Interpretation                     !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Message Type  ! Exch  ! Vers  !          Length               !
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                 SPI (unused)                                  !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                 SPI (unused)                                  !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                 Error Payload                                 !
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                 Sig/prf Payload
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The error message will contain the cookies as presented in the offending message, the message type OAKLEY_ERROR, and the reason for the error, followed by the rejected message.

Error messages are informational only, and the correctness of the protocol does not depend on them.

Error reasons:

```
TIMEOUT                         exchange has taken too long, state destroyed
AEH_ERROR                       an unknown algorithm appears in an offer
GROUP_NOT_SUPPORTED             GRP named is not supported
EXPONENTIAL_UNACCEPTABLE        exponential too large/small
SELECTION_NOT_OFFERED           selection does not occur in offer
NO_ACCEPTABLE_OFFERS            no offer meets host requirements
AUTHENTICATION_FAILURE          signature or hash function fails
RESOURCE_EXCEEDED               too many exchanges or too much state info
NO_EXCHANGE_IN_PROGRESS         a reply received with no request in progress
```

2.8 Additional Security for Privacy Keys: Private Groups

If the two parties have need to use a Diffie-Hellman key determination scheme that does not depend on the standard group definitions, they have the option of establishing a private group. The authentication need not be repeated, because this stage of the protocol will be protected by a pre-existing authentication key. As an extra security measure, the two parties will establish a private name for the shared keying material, so even if they use exactly the same group to communicate with other parties, the re-use will not be apparent to passive attackers.

Private groups have the advantage of making a widespread passive attack much harder by increasing the number of groups that would have to be exhaustively analyzed in order to recover a large number of session keys. This contrasts with the case when only one or two groups are ever used; in that case, one would expect that years and years of session keys would be compromised.

There are two technical challenges to face: how can a particular user create a unique and appropriate group, and how can a second party assure himself that the proposed group is reasonably secure?

The security of a modular exponentiation group depends on the largest prime factor of the group size. In order to maximize this, one can choose "strong" or Sophie-Germaine primes, P = 2Q + 1, where P and Q are prime. However, if P = kQ + 1, where k is small, then the strength of the group is still considerable. These groups are known as Schnorr subgroups, and they can be found with much less computational effort than Sophie-Germaine primes.

Schnorr subgroups can also be validated efficiently by using probable prime tests.

It is also fairly easy to find P, k, and Q such that the largest prime factor can be easily proven to be Q.

We estimate that it would take about 10 minutes to find a new group of about 2^1024 elements, and this could be done once a day by a scheduled process; validating a group proposed by a remote party would take perhaps a minute on a 25 MHz RISC machine or a 66 MHz CISC machine.

We note that validation is done only between previously mutually authenticated parties, and that a new group definition always follows and is protected by a key established using a well-known group. There are four points to keep in mind:

a. The description and public identifier for the new group are protected by the well-known group.

b. The responder can reject the attempt to establish the new group, either because he is too busy or because he cannot validate the largest prime factor as being sufficiently large.

c. The new modulus and generator can be cached for long periods of time; they are not security critical and need not be associated with ongoing activity.

d. Generating a new g^x value periodically will be more expensive if there are many groups cached; however, the importance of frequently generating new g^x values is reduced, so the time period can be lengthened correspondingly.

2.8.1 Defining a New Group

This section describes how to define a new group. The description of the group is hidden from eavesdroppers, and the identifier assigned to the group is unique to the two parties. Use of the new group for Diffie-Hellman key exchanges is described in the next section.

The secrecy of the description and the identifier increases the difficulty of a passive attack, because if the group descriptor is not known to the attacker, there is no straightforward and efficient way to gain information about keys calculated using the group.

Only the description of the new group need be encrypted in this exchange. The hash algorithm is implied by the OAKLEY session named by the group. The encryption is the authentication encryption function of the OAKLEY session.

The descriptor of the new group is encoded in the new group payload. The nonces are encoded in the Key Exchange Payload.

Data beyond the encryption boundary is encrypted using the transform named by the KEYID.

The following message use the ISAKMP Key Exchange Identifier OAKLEY New Group.

To define a new modular exponentiation group:

```
 Initiator                                       Responder
 ---------                                      ----------
  ->   KEYID,                                       ->
       INEWGRP,
       Desc(New Group), Na
       prf(sKEYID, Desc(New Group) | Na)

  <-   KEYID,
       INEWGRPRS,
       Na, Nb
       prf(sKEYID, Na | Nb | Desc(New Group))       <-

   ->  KEYID,
       INEWGRPACK
       prf(sKEYID, Nb | Na | Desc(New Group))        ->
```

These messages are encrypted at the encryption boundary using the key indicated. The hash value is placed in the "digital signature" field (see Appendix C).

New GRP identifier = Na | Nb (the initiator and responder must use nonces that are distinct from any cookies used for current KEYID's; i.e., the initiator ensures that Na is distinct from any CKY-I, the responder ensures that Nb is distinct from any CKY-R).

Desc(G) is the encoding of the descriptor for the group descriptor (see Appendix A for the format of a group descriptor)

The two parties must store the mapping between the new group identifier GRP and the group descriptor Desc(New Group). They must also note the identities used for the KEYID and copy these to the state for the new group.

Note that one could have the same group descriptor associated with several KEYID's. Pre-calculation of g^x values may be done based only on the group descriptor, not the private group name.

2.8.2 Deriving a Key Using a Private Group

Once a private group has been established, its group id can be used in the key exchange messages in the GRP position. No changes to the protocol are required.

2.9 Quick Mode: New Keys From Old,

When an authenticated KEYID and associated keying material sKEYID already exist, it is easy to derive additional KEYID's and keys, using only hashing functions. The KEYID might be one that was derived in Main Mode, for example.

On the other hand, the authenticated key may be a manually distributed key, one that is shared by the initiator and responder via some means external to OAKLEY. If the distribution method has formed the KEYID using appropriately unique values for the two halves (CKY-I and CKY-R), then this method is applicable.

In the following, the Key Exchange Identifier is OAKLEY Quick Mode. The nonces are carried in the Key Exchange Payload, and the prf value is carried in the Authentication Payload; the Authentication Authority is "None" and the type is "Pre-Shared".

The protocol is:

```
  Initiator                                           Responder
  ---------                                           ---------
  -> KEYID, INEWKRQ, Ni, prf(sKEYID, Ni)                ->
 <-  KEYID, INEWKRS, Nr, prf(sKEYID, 1 | Nr | Ni)      <-
  -> KEYID, INEWKRP, 0, prf(sKEYID,  0 | Ni | Nr)       ->
```

The New KEYID, NKEYID, is Ni | Nr

sNKEYID = prf(sKEYID, Ni | Nr)

The identities and EHA values associated with NKEYID are the same as those associated with KEYID.

Each party must validate the hash values before using the new key for any purpose.

2.10 Pre-Distributed Keys

If a key and an associated key identifier and state information have been distributed manually, then the key can be used for any OAKLEY purpose. The key must be associated with the usual state information: ID's and EHA algorithms.

Local policy dictates when a manual key can be included in the OAKLEY database. For example, only privileged users would be permitted to introduce keys associated with privileged ID's, an unprivileged user could only introduce keys associated with her own ID.

2.11 Distribution of an External Key

Once an OAKLEY session key and ancillary algorithms are established, the keying material and the "H" algorithm can be used to distribute an externally generated key and to assign a KEYID to it.

In the following, KEYID represents an existing, authenticated OAKLEY session key, and sNEWKEYID represents the externally generated keying material.

In the following, the Key Exchange Identifier is OAKLEY External Mode. The Key Exchange Payload contains the new key; the payload is protected using the encryption transform associated with the KEYID and associated key, sKEYID.

```
  Initiator                                                 Responder
  ---------                                                 ---------
  -> KEYID, IEXTKEY, Ni, prf(sKEYID, Ni)                           ->
 <-  KEYID, IEXTKEY, Nr, prf(sKEYID, 1 | Nr | Ni)                <-
  -> KEYID, IEXTKEY, Kir xor sNEWKEYID*, prf(Kir, sNEWKEYID | Ni | Nr) ->
```

Kir = prf(sKEYID, Ni | Nr)

* this field is carried in the first section of the Authentication Payload, encoded as a variable precision integer.

Each party must validate the hash values using the "H" function in the KEYID state before changing any key state information.

The new key is recovered by the Responder by calculating the xor of the field field in the Authentication Payload with the Kir value.

The new key identifier, naming the keying material sNEWKEYID, is prf(sKEYID, 1 | Ni | Nr).

Note that this exchange does not require encryption. Hugo Krawcyzk suggested the method and its advantage.

2.12. Cryptographic Strength Considerations

The strength of the key used to distribute the external key must be at least equal to the strength of the external key. Generally, this means that the length of the sKEYID material must be greater than or equal to the length of the sNEWKEYID material.

The derivation of the external key, its strength or intended use are not addressed by this protocol; the parties using the key must have some other method for determining these properties.

As of early 1996, it appears that for 90 bits of cryptographic strength, one should use a modular exponentiation group modulus of 2000 bits. For 128 bits of strength, a 3000 bit modulus is required.

3. Specifying and Deriving Security Associations

When a security association is defined, only the KEYID need be given. The responder should be able to look up the state associated with the KEYID value and find the appropriate keying material, sKEYID.

The OAKLEY protocol does not define security association encodings or message formats. These can be defined through a protocol such as ISAKMP. Compatibility with ISAKMP is a goal of the OAKLEY design, and coordination of the message formats and use of identifiers is an ongoing activity at this time.

4. ISAKMP Compatibility

OAKLEY uses ISAKMP header and payload formats, as described in the text and in Appendix B. There are particular noteworthy extensions beyond the version 4 draft.

4.1 Authentication with Existing Keys

In the case that two parties do not have suitable public key mechanisms in place for authenticating each other, they can use keys that were distributed manually. After establishment of these keys and their associated state in OAKLEY, they can be used for authentication modes that depend on signatures, e.g. Aggressive Mode.

When an existing key is to appear in an offer list, it should be indicated with an Authentication Authority of ISAKMP_EXISTING. The value for this field must not conflict with any authentication authority registered with IANA.

The authentication payload will have two parts:
    the KEYID for the pre-existing key

    the identifier for the party to be authenticated by the pre-
   existing key.

The pseudo-random function "H" in the state information for that KEYID will be the signature algorithm, and it will use the keying material for that key (sKEYID) when generating or checking the validity of message data.

E.g. if the existing key has an KEYID denoted by KID and 128 bits of keying material denoted by sKID and "H" algorithm a transform named HMAC, then to generate a "signature" for a data block, the output of
HMAC(sKID, data)
will be the corresponding signature payload.

The KEYID state will have the identities of the local and remote parties for which the KEYID was assigned; it is up to the local policy implementation to decide when it is appropriate to use such a key for authenticating other parties. For example, a key distributed for use between two Internet hosts A and B may be suitable for authenticating all identities of the form "alice@A" and "bob@B".

4.2 Third Party Authentication

A local security policy might restrict key negotiation to trusted parties. For example, two OAKLEY daemons running with equal sensitivity labels on two machines might wish to be the sole arbiters of key exchanges between users with that same sensitivity label. In this case, some way of authenticating the provenance of key exchange requests is needed. I.e., the identities of the two daemons should be bound to a key, and that key will be used to form a "signature" for the key exchange messages.

The Signature Payload, in Appendix B, is for this purpose. This payload names a KEYID that is in existence before the start of the current exchange. The "H" transform for that KEYID is used to calculate an integrity/authentication value for all payloads preceding the signature.

Local policy can dictate which KEYID's are appropriate for signing further exchanges.

4.3 New Group Mode

OAKLEY uses a new KEI for the exchange that defines a new group.

5. Security Implementation Notes

Timing attacks that are capable of recovering the exponent value used in Diffie-Hellman calculations have been described by Paul Kocher [Kocher]. In order to nullify the attack, implementors must take pains to obscure the sequence of operations involved in carrying out modular exponentiations.

A "blinding factor" can accomplish this goal. A group element, r, is chosen at random. When an exponent x is chosen, the value r^(-x) is also calculated. Then, when calculating (g^y)^x, the implementation will calculate this sequence:

```
        A = (rg^y)
        B = A^x = (rg^y)^x = (r^x)(g^(xy))
        C = B*r^(-x) = (r^x)(r^-(x))(g^(xy)) = g^(xy)
```

The blinding factor is only necessary if the exponent x is used more than 100 times (estimate by Richard Schroeppel).

6. OAKLEY State Machine

There are many pathways through OAKLEY.

The initiator decides on an initial message in the following order:

1. Offer a cookie. This is not necessary but it helps with aggressive exchanges.

2. Pick a group. The choices are the well-known groups or any private groups that may have been negotiated. The very first exchange between two Oakley daemons with no common state must involve a well-known group.

3. Pick an exponent x and present g^x.

4. Offer Encryption, Hash, and Authentication lists.

5. Use PFS for hiding the identities

If identity hiding is not used, then the initiator has this option:

6. Name the identities and include authentication information

The information in the authentication section depends on the first authentication offer. In this aggressive exchange, the Initiator hopes that the Responder will accept all the offered information and the first authentication method. The authentication method determines the authentication payload as follows:

1. A signing method will be used to sign all the offered information.

2. A public key encryption method will be used to encrypt a nonce in the public key of the requested reponder identity. There are two cases possible, depending on whether or not identity hiding is used:

   a. No identity hiding. The ID's will appear as plaintext.
   b. Identity hiding. A well-known ID, call it R', will appear as plaintext in the authentication payload. It will be followed by two ID's and a nonce; these will be encrypted using the public key for R'.

3. A pre-existing key method. The pre-existing key will be used to encrypt a nonce. If identity hiding is used, the ID's will be encrypted in place in the payload, using the "E" algorithm associated with the pre-existing key.

The responder can accept all, part or none of the initial message.

The responder accepts as many of the fields as he wishes, using the same decision order as the initiator. At any step he can stop, implicitly rejecting further fields. The minimum response is a cookie and the GRP.

1. Accept cookie. The responder may elect to record no state information until the initiator successfully replies with a cookie chosen by the responder. If so, the responder replies with a cookie, the GRP, and no other information.

2. Accept GRP. If the group is not acceptable, the responder will not reply. The responder may send an error message indicating the the group is not acceptable (modulus too small, unknown identifier, etc.)

3. Accept the g^x value. The responder indicates his acceptance of the g^x value by including his own g^y value in his reply. He can postpone this by ignoring g^x and putting a zero length g^y value in

his reply.

4. Accept one element from each of the EHA lists. The acceptance is indicated by a non-zero proposal.

5. If PFS for identity hiding is requested, then no further data will

follow.

6. If the authentication payload is present, and if the first item in the offered authentication class is acceptable, then the responder must should validate/decrypt the information in the authentication payload and hash payload, if present. The responder should choose a nonce and reply using the same authentication/hash algorithm as the initiator used.

APPENDIX A Group Descriptors

Three distinct group representations can be used with OAKLEY. Each group is defined by its group operation and the kind of underlying field used to represent group elements. The three types are modular exponentiation groups (named MODP herein), elliptic curve groups over the field GF[2^N] (named EC2N herein), and elliptic curve groups over GF[P] (named ECP herein) For each representation, many distinct realizations are possible, depending on parameter selection.

With a few exceptions, all the parameters are transmitted as if they were non-negative multi-precision integers, using the format defined in this appendix (note, this is distinct from the encoding in Appendix D). Every multi-precision integer has a prefixed length field, even where this information is redundant.

For the group type EC2N, the parameters are more properly thought of as very long bit fields, but they are represented as multi-precision integers, (with length fields, and right-justified). This is the natural encoding.

MODP means the classical modular exponentiation group, where the operation is to calculate G^X (mod P). The group is defined by the numeric parameters P and G. P must be a prime. G is often 2, but may be a larger number. 2 <= G <= P-2.

ECP is an elliptic curve group, modulo a prime number P.
The defining equation for this kind of group is
 Y^2 = X^3 + AX + B
The group operation is taking a multiple of an elliptic-curve point.
The group is defined by 5 numeric parameters: The prime P, two curve parameters A and B, and a generator (X,Y). A,B,X,Y are all interpreted mod P, and must be (non-negative) integers less than P.
They must satisfy the defining equation, modulo P.

EC2N is an elliptic curve group, over the finite field F[2^N]. The defining equation for this kind of group is
 Y^2 + XY = X^3 + AX^2 + B
(This equation differs slightly from the mod P case: it has an XY term, and an AX^2 term instead of an AX term.)

We must specify the field representation, and then the elliptic curve. The field is specified by giving an irreducible polynomial (mod 2) of degree N. This polynomial is represented as an integer of size between 2^N and 2^(N+1), as if the defining polynomial were evaluated at the value U=2.

For example, the field defined by the polynomial
U^155 + U^62 + 1
is represented by the integer 2^155 + 2^62 + 1. The group is defined by 4 more parameters, A,B,X,Y. These parameters are elements of the field F[2^N], and can be though of as polynomials of degree < N, with

```
   (mod 2) coefficients.  They fit in N-bit fields, and are represented
   as integers < 2^N, as if the polynomial were evaluated at U=2.  For
   example, the field element U^2 + 1 would be represented by the
   integer 2^2+1, which is 5.  The two parameters A and B define the
   curve.  A is frequently 0.  B must not be 0.  The parameters X and Y
   select a point on the curve.  The parameters A,B,X,Y must satisfy the
   defining equation, modulo the defining polynomial, and mod 2.

   Group descriptor formats:

    Type of group: A two-byte field,
           assigned values for the types "MODP", "ECP", "EC2N"
           will be defined (see ISAKMP-04).
   Size of a field element, in bits.  This is either Ceiling(log2 P)
      or the degree of the irreducible polynomial: a 32-bit integer.
   The prime P or the irreducible field polynomial: a multi-precision integer.
   The generator: 1 or 2 values, multi-precision integers.
   EC only:  The parameters of the curve:  2 values, multi-precision integers.

   The following parameters are Optional (each of these may appear
   independently):
     a value of 0 may be used as a place-holder to represent an unspecified
     parameter; any number of the parameters may be sent, from 0 to 3.

   The largest prime factor: the encoded value that is the LPF of the group size,
     a multi-precision integer.

   EC only:  The order of the group: multi-precision integer.
     (The group size for MODP is always P-1.)

   Strength of group: 32-bit integer.
     The strength of the group is approximately the number of key-bits protected.
     It is determined by the log2 of the effort to attack the group.
     It may change as we learn more about cryptography.

This is a generic example for a "classic" modular exponentiation group:
      Group type: "MODP"
      Size of a field element in bits:  Log2 (P) rounded *up*.  A 32bit integer.
      Defining prime P: a multi-precision integer.
      Generator G: a multi-precision integer.  2 <= G <= P-2.
      <optional>
      Largest prime factor of P-1: the multi-precision integer Q
      Strength of group: a 32-bit integer.  We will specify a formula
        for calculating this number (TBD).

   This is a generic example for elliptic curve group, mod P:
      Group type: "ECP"
      Size of a field element in bits:  Log2 (P) rounded *up*,
```

a 32 bit integer.
Defining prime P: a multi-precision integer.
Generator (X,Y): 2 multi-precision integers, each < P.
Parameters of the curve A,B: 2 multi-precision integers, each < P.
<optional>
Largest prime factor of the group order: a multi-precision integer.

Order of the group: a multi-precision integer.
Strength of group: a 32-bit integer. Formula TBD.

This is a specific example for elliptic curve group:

Group type: "EC2N"
Degree of the irreducible polynomial: 155
Irreducible polynomial: U^155 + U^62 + 1, represented as the multi-precision integer 2^155 + 2^62 + 1.
Generator (X,Y) : represented as 2 multi-precision integers, each < 2^155. For our present curve, these are (decimal) 123 and 456. Each is represented as a multi-precision integer.
Parameters of the curve A,B: represented as 2 multi-precision integers, each < 2^155. For our present curve these are 0 and (decimal) 471951, represented as two multi-precision integers.

<optional>
Largest prime factor of the group order:

38059938472158930161554638261953862663974364443,

represented as a multi-precision integer.
The order of the group:

45671926166590716193865565914344635196769237316

represented as a multi-precision integer.

Strength of group: 76, represented as a 32-bit integer.

The variable precision integer encoding for group descriptor fields is the following. This is a slight variation on the format defined in Appendix D in that a fixed 16-bit value is used first, and the length is limited to 16 bits. However, the interpretation is otherwise identical.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   Fixed value (TBD)           !             Length            !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                  Integer                                      .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of a group descriptor is:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!1!     Group Description     !             MODP              !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!0!        Field Size         !            Length             !
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!0!          Prime           !            Length                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!0!       Generator1         !            Length                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!0!       Generator2         !            Length                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!0!         Curve-p1         !            Length                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!0!         Curve-p2         !            Length                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!0!   Largest Prime Factor   !            Length                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!1!0!      Order of Group      !            Length                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!0!0!    Strength of Group     !            Length                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                              MPI                                !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

APPENDIX B Message formats

1. The ISAKMP Message Types and Header

OAKLEY uses the ISAKMP Message Types ISA_KE&AUTH_REQ and ISA_KE&AUTH_REP for all key exchanges.

OAKLEY uses the two SPI fields in an ISAKMP header for the name of the abstract group (GRP) used in the negotiation. If no group is used (forfeiting perfect forward secrecy), then the field will have the value 0.

```
                         1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    !                                                               !
```

```
              ~                       Initiator-Cookie                  ~
           /  !                                                         !
     KEYID    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           \  !                                                         !
              ~                       Responder-Cookie                  ~
              !                                                         !
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              !                  Domain of Interpretation               !
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              ! Message Type  ! Exch  ! Vers  !          Length         !
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              !                 Group ID                                !
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              !                 Group ID                                !
         eeee +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ eeee
              !                 ...                                     !
```

"eeee" represents the encryption boundary for messages requiring privacy. The message after this point is subject to the encryption transform implied by the KEYID.

The Group ID field is used for the group identifier for the key exchange methods described in this document; in other ISAKMP messages the field is used for a SPI.

The second SPI field is not used in OAKLEY. It must contain the value zero.

The OAKLEY proposal format contains the SA attributes that are exchanged in the ISA_INIT messages in order to establish the required security at- tributes for the key and authentication exchange. See [OAKLEY] for fur- ther details.

2. OAKLEY Use of ISA_AUTH&KE packets.

```
                                  1                   2                   3
               0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              ~                        ISAKMP Header                    ~
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              ! Next Payload  !  Payload Len  !           RESERVED      !
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              ~                                                         ~
              !                   Authentication Payload                !
```

```
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !  Payload Len  !            RESERVED           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
!                     Key Exchange Payload                      !
```

```
       ~                                                               ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ISA_AUTH&KE_REQ and ISA_AUTH&KE_RESP Packet Format

The encodings of the OAKLEY parameters into these fields are described in the next sections.

3. The Key Exchange Payload

The Key Exchange Payload carries values that are used to derive secret keying material. Because OAKLEY uses both nonces and Diffie-Hellman exponentials for deriving keys, its use of the Key Exchange Payload is slightly different from the use described in ISAKMP; that document expects only one Key Exchange Payload per packet, but OAKLEY can have two, one for nonces, one for an exponential.

```
                            1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ! Next Payload  !  Payload Len  !            RESERVED           !
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       !                  KEI          !            Length             !
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       ~                                                               ~
       !                       Key Exchange Data                       !
       ~                                                               ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Key Exchange Payload Format

- o KEI (2 octets) - Key Exchange Identifier
- o Length (2 octets) - Length of payload in octets
- o Key Exchange Data (variable) - Data required to create session key.

OAKLEY uses three KEI values: OAKLEY Main Mode, OAKLEY Quick Mode, OAKLEY External Mode, OAKLEY New Group Mode.

The value encoded in the Key Exchange Data field will be the Diffie-Hellman exponential (if it is used), encoded as variable precision integers as shown in Appendix D.

3. The OAKLEY Authentication Payload

```
                      1                   2                   3
```

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !  Payload Len  !            RESERVED           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Authentication Authority     !           Reserved            !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Authentication Type          !            Length             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
!                     Authentication Data                       !
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Authentication Payload Format

The Authentication Payload will be used to carry three pieces of essential information: the entity identifiers (ID's), the nonces, and the output of a function proving proving knowledge of a secret.

The format of the ID's is described in the next section. A payload will have two ID's, for the Initiator and Responder, in that order. If the length of an ID is zero, the ID is unspecified.

If the low order bit of the Reserved field is set, the payload will have three ID's; see section 2.4.2, An Aggressive Example With Hidden Identities.

The nonce will follow the ID's; if a nonce is zero length, it is considered to be not present.

The fourth part of the authentication payload will contain the result of applying the pseudorandom function or signature algorithm to the key exchange parameters, as described in the main text. For example, the output might be the result of applying a keyed MD5 transform to the ID's, the cookies, the nonces, and the exponentials.

The pseudorandom function output will encoded as a variable precision integer as described in Appendix D.

4. The OAKLEY Proposal

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!    OAKLEY     !  Proposal #   !  Proposal Len !    RESERVED   !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
!                             EHA  Format                           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                            Group Format                           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

OAKLEY Proposal Format

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !0!1!     Auth / Priv  Flag     !            PRIV               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !0!1!   Encryption Algorithm    !             DES               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !0!1!       Hash Algorithm      !             MD5               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !1!1!    Authentication Alg     !             RSA               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !0!1!    Authentication Mode    !            KEYED              !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

OAKLEY Proposal - EHA Format

5. Identity (ID) formats

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !  Identification Type          !            Length             !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ~                                                               ~
 !                    Identification Data                        !
 ~                                                               ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

There are three identification types: IP_ADDR (value 1), FQDN (value 2), USER_FQDN (value 3).

The length of the IP address will be 4 bytes for the IPv4 Domain of Interpretation, 8 bytes for the IPv6 DOI.

FQDN is a fully qualified domain name, as used by the DNS protocol. Its form is an ASCII character string. The domain components are separated by "." characters, as in DNS.

USER_FQDN is a user id followed by a "." character, followed by a fully qualified domain name, as used by the DNS protocol. Its form is an ASCII character string.

6. OAKLEY's use ISA_INIT_REQ and ISA_INIT_RESP Packets

OAKLEY does not require the use the ISAKMP ISA_INIT_REQ and ISA_INIT_RESP packets. Their optional use may include the establishment of ISAKMP-to-ISAKMP daemon KEYID's for later use as signatures over ISA_KE&AUTH packets, providing an extra level of authenticity checking. In this case, the Situation field will have the IP addresses of the two principals; the length of the IP address will depend on the Domain of Interpretation.

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ~                          ISAKMP Header                        ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !  Payload Len  !           RESERVED            !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                  Domain of Interpretation                     !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~                           Situation                           ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~                           Proposal                            ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ISA_INIT_REQ and ISA_INIT_RESP Packet Format

7. Digital Signature/PRF Payload

The Digital Signature/PRF payload will carry a value for authenticating the entire message. When it occurs, it will be the last payload.

```
                    1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                            KEYID                              !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ~                                                               ~
 !                    Signature/hash data                        !
 ~                                                               ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The output of the signature or prf function will be encoded as a variable precision integer as described in Appendix D. The KEYID will indicate KEYID that names keying material and the Hash or Signature function.

8. The Credential Payload

Useful certificates with public key information can be attached to

OAKLEY messages using Credential Payloads. The format of the payload depends on the Authentication Type, and separate RFC's define the formats. The encoding of the Authority and Type are the same as for the Authentication Payload.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! Next Payload  !  Payload Len  !            RESERVED           !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  Authentication Authority     !           Reserved            !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  Authentication Type          !            Length             !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                                                               ~
   !                      Credential Data                          !
   ~                                                               ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Credential Payload Format

APPENDIX D Encoding a variable precision integer.

Variable precision integers will be encoded as a 32-bit length field followed by one or more 32-bit quantities containing the representation of the integer, aligned with the most significant bit in the first 32-bit item.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!    length                                                     !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!    first value word (most significant bits)                   !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~     additional value words                                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An example of such an encoding is given below, for a number with 51 bits of significance. The length field indicates that 2 32-bit quantities follow. The most significant non-zero bit of the number is in bit 13 of the first 32-bit quantity, the low order bits are in the second 32-bit quantity.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                            1 0!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!0 0 0 0 0 0 0 0 0 0 0 0 0 1 x x x x x x x x x x x x x x x x x x!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

APPENDIX E Cryptographic strengths

The Diffie-Hellman algorithm is used to compute keys that will be used with symmetric algorithms. It should be no easier to break the Diffie-Hellman computation than it is to do an exhaustive search over the symmetric key space. A recent recommendation by an group of cryptographers [Blaze-Diffie-et-al] has recommended a symmetric key size of 75 bits for a practical level of security. For 20 year security, they recommend 90 bits.

Based on that report, a conservative strategy for OAKLEY users would be to ensure that their Diffie-Hellman computations were as secure as at least a 90-bit key space. In order to accomplish this for modular exponentiation groups, the size of the largest prime factor of the modulus should be at least 180 bits, and the size of the modulus should be at least 1400 bits. For elliptic curve groups, the LPF should be at least 180 bits.

If long-term secrecy of the encryption key is not an issue, then the following parameters may be used for the modular exponentiation group: 150 bits for the LPF, 980 bits for the modulus size.

The modulus size alone does not determine the strength of the Diffie-Hellman calculation; the size of the exponent used in computing powers within the group is also important. The size of the exponent in bits should be at least twice the size of any symmetric key that will be derived from it. We recommend that ISAKMP implementors use at least 180 bits of exponent (twice the size of a 20-year symmetric key).

The mathematical justification for these estimates can be found in texts that estimate the effort for solving the discrete log problem, a task that is strongly related to the efficiency of using the Number Field Sieve for factoring large integers. Readers are referred to [Stinson] and [Schneier].

APPENDIX F The Well-Known Groups

This section will have explicit descriptors for three modular exponentiation groups and two elliptic curve over GF[2^n] groups.

The identifiers for the groups (the well-known GRP's) will also be given here.

```
0   No group (used as a placeholder and for non-DH exchanges)
1   A modular exponentiation group with a 768 bit modulus (TBD)
2   A modular exponentiation group with a 1024 bit modulus (TBD)
3   A modular exponentiation group with a 2048 bit modulus (TBD)
4   An elliptic curve group over GF[2^155]
5   An elliptic curve group over GF[2^210]
```

values 2^32 and higher are used for private group identifiers

Appendix K Implementing Group Operations

The group operation must be implemented as a sequence of arithmetic operations; the exact operations depend on the type of group. For modular exponentiation groups, the operation is multi-precision integer multiplication and remainders by the group modulus. See Knuth Vol. 2 [Knuth] for a discussion of how to implement these for large integers. Implementation recommendations for elliptic curve group operations over GF[2^N] are described in [Schroeppel].

BIBLIOGRAPHY

[RFC1825] Atkinson, Randall, RFC's 1825-1827

[Blaze] Blaze, Matt et al., Recent symmetric key report

[STS] Diffie, van Oorschot, and Wiener, Authentication and Authenticated Key Exchanges

[DSS] DSS draft-ietf-ipsec-dss-cert-00.txt

[SECDNS] DNS Signed Keys, Eastlake & Kaufman, draft-ietf-dnssec-secext-09.txt

[Photuris] Karn, Phil and Simpson, William, Photuris, draft-ietf-ipsec-photuris-09.txt

[Kocher] Kocher, Paul, Timing Attack

[Krawcyzk] Krawcyzk, Hugo, SKEME, ISOC, SNDS Symposium, San Diego, 1996

[PKIX] PKIX internet drafts, draft-ietf-pkix-ipki-00.txt

[Random] Random number RFC 1750

[ISAKMP] Schertler, Mark, ISAKMP, draft-ietf-ipsec-isakmp-03.txt and draft-ietf-ipsec-isakmp-04.txt. The transition from version 3 to version 4 was in progress at the time of this writing.

[Schneier] Schneier, Bruce, Applied cryptography: protocols, algorithms, and source code in C, Second edition, John Wiley & Sons, Inc. 1995, ISBN 0-471-12845-7, hardcover. ISBN 0-471-11709-9, softcover.

[Schroeppel] Schroeppel, Richard, et al.; Fast Key Exchange with Elliptic Curve Systems, Crypto '95, Santa Barbara, 1995. Available on-line as ftp://ftp.cs.arizona.edu/reports/1995/TR95-03.ps (and .Z).

[Stinson] Stinson, Douglas, Cryptography Theory and Practice. CRC Press, Inc., 2000, Corporate Blvd., Boca Raton, FL, 33431-9868, ISBN 0-8493-8521-0, 1995

[Zimmerman] Philip Zimmermann, The Official Pgp User's Guide, Published by MIT Press Trade, Publication date: June 1995, ISBN: 0262740176

This draft expires six months from the day of issue. The expiration date will be August 24, 1996.