IPSEC Working Group H. K. Orman
INTERNET-DRAFT Dept. of Computer Science, Univ. of Arizona
draft-ietf-ipsec-oakley-00.txt February 1996

The Oakley Key Determination Protocol
<draft-ietf-ipsec-oakley-00.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.

This 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

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 this 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 doing key exchange first and encrypted authentication next, but it extends the STS protocol in five ways.

The first is the addition of a weak authentication 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 protocol provides Perfect Forward Secrecy (PFS) in its standard mode of operation; with PFS, the security of the shared key against passive attacks is not dependent on other any other secret.

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 modes for distributing keys. In addition to the classic Diffie-Hellman exchange, this protocol has a mode of use for deriving a new key from a pre-existing key, and a mode for distributing an externally derived key by encrypting it.

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. Subsequent stages of the protocol may derive other keys from a named key and assign an identifier to the new key. 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 anciliary algorithms for OAKLEY; their appearance in subsequent security association definitions derived with other protocols is neither required nor prohibited.

The anti-clogging tokens, or "cookies", provide a weak form of authentication for both parties; the cookie exchange can be completed before they must perform the computationally expensive part of the protocol (the 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".

The only 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 can be used directly over the IP protocol as protocol id [TBD] or over the UDP

protocol. In the latter case, the only addressing requirement is that protocol exchanges be initiated by using the OAKLEY well-known port [TBD] in the destination address.

The machine running OAKLEY must provide a good random number generator, as described in RFCxxxx, 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.

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, IREQ, g^x                      ->
         <-    Cookie-R, Cookie-I, IRES, 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 IREQ, 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 IRES, and the number g raised to the y'th power.

The values IRES and IREQ are in capitals to indicate that they are unique constants (constants are defined the appendices).

2.2.2 Guide to symbols

Cookie-I and Cookie-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.

DOI is the Domain of Interpretation; see appendix I. The domains are statically assigned numbers that indicate classes of cryptographic service -- particularly the strength of the algorithm.

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.

IREQ, IREP, IKERQ, and IKERS are distinct message identifiers.

Auth/Priv (or A/P) is the encoded choice for the intended use of the keying material; either authentication or privacy.

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 Appenix 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 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 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.

2.3 Main Mode

The goal of Main Mode processing is to establish common state in the two parties. This state information is a key name, secret keying material, and three algorithms for use during authentication:

encryption, hashing, and authentication. The encodings and meanings for these choices are presented in an Appendix.

Main Mode processing is always followed by Authentication, as described in the Authentication Exchange section. However, see also the section on use of Main Mode with private group definitions.

```
Initiator                                                  Responder
           -> Cookie-I, 0, DOI, IREQ, A/P, GRP, EHAO          ->
         <-  Cookie-R, Cookie-I, DOI, IKREQ, A/P, g^y, EHAS   <-
          -> Cookie-I, Cookie-R, DOI, IKRES, g^x              ->
```

The processing outline for each stage of the protocol 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: DOI, Auth/Priv, GRP, EHAO list,

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

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

The responder receives IREQ and does the following:

Generates a unique cookie, Cookie-R,

associates the triple (Cookie-I, Cookie-R, DOI) with the Auth/Priv choice, the group GRP and the IP address of the responder, and

puts the network address of the message into the state and,

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

selects one algorithm from each class in the EHAO (encryption-hash-authentication algorithm offers) list, and

selects a g^y value and associates it with the current state, and

sets a timer for possible retransmission, and

sends the IKREQ message.

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

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

adds g^y to its state information,

chooses an exponent x and corresponding g^x value,

saves the EHA selections in the state,

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

sends the IKRES message.

The responder receives the IKRES message and
validates the Cookie pair against the network address for the incoming messages,

computes (g^y)^x (= g^yx = g^xy).

The responder can upgrade the initiator's A/P choice from Authentication to Privacy; the initiator must cooperate.

If privacy is a requirement, then encryption in the algorithm indicated by the EHA class will affect subsequent messages of the exchange. The cookies and message type word will be the only non-

encrypted part of those messages (see Appendix Message Formats for the encryption boundary).

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.

Both parties compute the shared key material sKEYID as
hash(g^xy | Cookie-I | Cookie-R)
where "hash" is the algorithm in class "hash" selected in the EHA list.

The initiator considers the ability of the responder to repeat Cookie-I as weak evidence that the message originates from a "live" correspondent on the network and the correspondent is associated with the responder's network address. The responder makes similar assumptions when Cookie-R is repeated to the responder. All messages except IREQ messages must have valid cookies; information in violating messages cannot be used for any OAKLEY operations.

2.3.1 Retransmission, Timeouts, and Error Messages

Retransmissions due to failure to elicit an expected response in the appropriate time interval must be handled gracefully by both parties.

Either party may destroy the current state and optionally send an error message at any point in the protocol.

The responder can explicitly reject the initial request for several

reasons: no support for a well-known but optional group, a malformed EHA list, or a temporary lack of resources, for example. The exact format for error messages is TBD.

2.3.2 ISAKMP Compatibility

In addition to Main Mode, this document describes three other key determination methods. Each method is intended to constitute a Key

Exchange Interface (KEI) that could be used with ISAKMP; each method also constitutes a protocol that could be used independently from ISAKMP.

The next method is described in order to establish an exact correspondence with the initial processing of ISAKMP in draft 03; the cookie exchange and the g^x exchanges are done as separate steps. This orthogonality may be desirable to some implementors, and it is thus included as a required OAKLEY mode.

2.3.2.1 ISAKMP Cookie/KE Mode

The ISAKMP protocol draft [ISAKMP-03] separates the cookie exchange entirely from the exchange of group elements. This is also allowable in OAKLEY. The following table illustrates the message sequence and fields roughly as described in ISAKMP-03. This sequence uses notation from the ISAKMP-03 draft and is included for merely to illustrate how Oakley and ISAKMP can be closely related to each other. A fuller treatment will appear later.

```
Initiator                                              Responder
---------                                             ----------
     -> Cookie-I, 0, IREQI, SPI-I                          ->
    <-  Cookie-R, Cookie-I, IRESI, SPI-R                  <-
     -> Cookie-I, Cookie-R, IKREQI, SPI-I, g^x, EHAO       ->
    <-  Cookie-R, Cookie-I, IKRESI, SPI-R, g^y, EHAS      <-
```

For compatibility with ISAKMP, the following message type value equivalences are required:
IREQI == ISA_INIT_REQ
IRESI == ISA_INIT_RESP
IKREQI == ISA_KE_REQ
IKRESI == ISA_KE_RESP
Note that the ISAKMP version 3 uses the GRPID field for a SPI field. Appendix C shows the correspondence of fields.

Fields that are not mentioned in the message summaries above are must contain the value zero.

2.4 Authentication

After the shared keying material and its identifier are established in Main Mode, the two parties must establish their identities to each other. The keying material cannot be used for any trusted purpose until the authentication is completed. If the purpose of the keying material is for encryption, then the identities of the initiator and responder will be hidden by encrypting the messages using the algorithm selected from the encryption clas in the EHAO list.

The authentication exchange not only hides the identities of the two parties, but it also avoids using public key technology that would provide a proof, verifiable by a third party, of communication between the initiator and responder. This technique and its justification are due to [Krawcyzk].

2.4.1 The Authentication Exchange

The Authentication Exchange should be initiated after Main Mode. The purpose of it is to change the state of KEYID from Unauthenticated to Authenticated. When using Main Mode with a well-known group, The authentication MUST be completed before using the keying material for any purpose, other than described in this section.

The authentication sequence binds the identities of the two parties to the KEYID. However, for most authentication methods, there will be two further steps: retrieving the material that describes the binding between the identity and a public key (e.g. a certificate), and validating that material (e.g. verifying the signature of the certifying authority). This section describes the binding to the KEYID; subsequent sections discuss the formats for the descriptive material and the retrieval methods.

The Authentication Exchange is carried out in the following classic Needham-Schroeder style:

```
Initiator                                                    Responder
---------                                                   ----------
  ->  KEYID, IAUTH_REQ, ID[A]                                    ->
 <-   KEYID, IAUTH_RES, ID[B], ID[A],
       E[Nb]KA, hash(sKEYID | Nb | ID[A] | ID[B])                <-
  ->  KEYID, IAUTH_PRF, E[Na]KB, hash(sKEYID | Na | Nb | ID[A],ID[B]) ->
 <-   KEYID, IAUTH_PRF_R, hash(sKEYID | Nb | Na | ID[B] | ID[A])     <-
```

The symbol ID[A] means the encoding of the identity of the initiator, the symbol ID[B] is for the responder. See the next section for a

discussion of the encoding of the identities.

The notation E[Nb]KA means the encryption of the nonce supplied by the responder encrypted using the key belonging to the initiator. When public key technology is used for authentication, this encryption is done using the public key encryption algorithm. If symmetric keys are used, the encryption is done in the symmetric

algorithm.

The encryption of the nonces is carried out in addition to the encryption described next. The nonce encryption is used to validate identities; the privacy encryption is to prevent disclosure of the purported identities of the two parties.

If the Auth/Priv type of KEYID is Privacy, then these messages are encrypted using the algorithm implied by the KEYID. The encryption boundary is shown in Appendix C. The KEYID and Message Type word are in cleartext.

The encryption of the nonces Na and Nb is done with the privacy algorithm established for the keyid. The key used in the encryption varies, depending on the authentication type selected during the Main Mode. If pre-shared keys are used, then the encryption is done with the pre-shared key. If public keys are used, then the public key of the opposite party is used.

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

After the Authentication Exchange is complete, the state of the KEYID is changed from Unauthenticated to Authenticated, and the keying material is ready for use.

2.4.1.1 Extra Strength for Protection of Encryption Keys

If the Auth/Priv type associated with KEYID is Privacy, then after the authentication exchange is complete, the nonces Na and Nb are used to provide an extra dimension of secrecy in deriving session keys. They are used as input to a hash function that derives the

keying material:

```
sKEYID <- hash(sKEYID | Na | Nb)
```

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.

2.4.2 Formats of Identity Data Structures

At this time, there is no commonly accepted basis for determining identities on the Internet, so the protocol must maintain room for flexibility on this point. There will be the following possibilities:

a. Pre-shared symmetric keys
Each pair of parties has arranged for a trusted method of

distributing secret keys for their mutual 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. See Quick Mode.

b. 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 is described in Appendix PGP. Support for this is OPTIONAL.

c. 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.

i. The format for X.509 OAKLEY certificates is described in Appendix X.509. Support for this is OPTIONAL.

d. DSS keys w/ certificates

Encoding for the Digital Signature Standard is described in Appendix H and in internet-draft-dss-00.txt.

2.4.2 Validating Identity Data Structures

The combination of the Authentication algorithm defines how to interpret the identity, its certificate, and the preferred method for fetching the cerificate if it is not included as part of the identity in the authentication exchange.

Once the certificate is obtained (see 2.4.3), the validation method will depend on the Authentication algorithm; 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.

At this time there is no required format or validation method.

2.4.3 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 retrieval method will be specified via an implicit attribute of the Auth class name.

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.

2.5 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 encryption. 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.

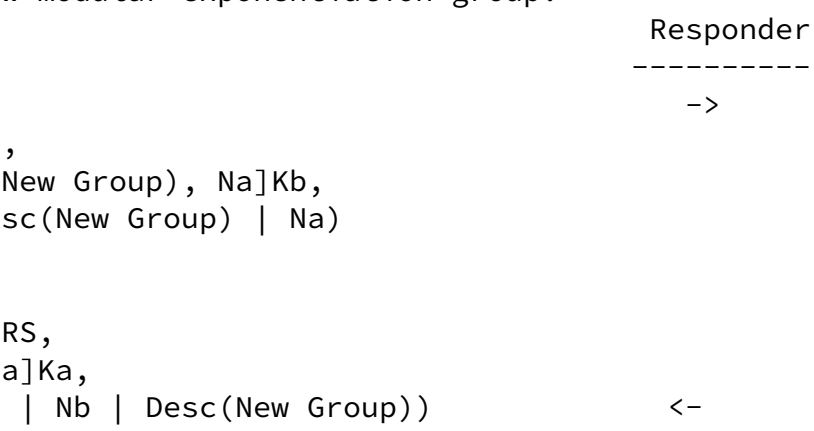
2.5.1 Defining a New Group

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.

To define a new modular exponentiation group:



```
   ->  KEYID,
       INEWGRPACK
       hash( 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).

INEWGRP, INEWGRPRS, INEWGRPACK are distinct message identifiers

Kb is the authentication key for B

Ka is the authentication key for A

These keys are derived during the authentication phase and are part of the state associated with the OAKLEY session named by the cookies.

E[x]Ka indicates encryption of x using the initiator's key; Kb indicates the responder's key (if the encryption algorithm is symmetric one the keys will be identical).

If Ka and Kb are public keys, then encryption will use the algorithm implied by the public key scheme, i.e., RSA encryption for RSA public keys.

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 Cookie-I, the responder ensures that Nb is distinct from any Cookie-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.

2.6 Deriving a Key Using a Private Group

Once a private group has been established, its group id can be used in Main Mode (or ISAKMP mode) to derive new keying material.

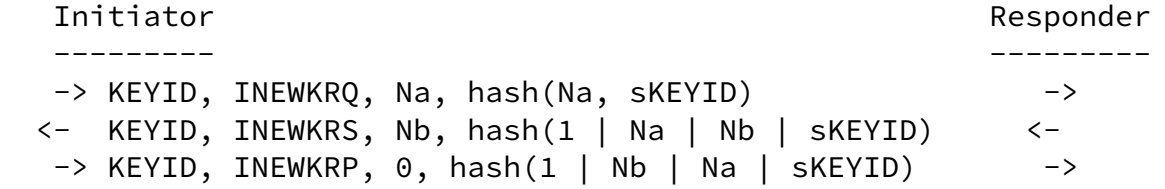
The authentication exchange is unnecessary, because the new group establishment was done using an authenticated key. The identities used in that exchange must be carried over to new key.

2.7 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 the distribution method has formed the KEYID using appropriately unique values for the two halves (Cookie-I and Cookie-R), then this method is applicable.

The protocol is:



The New KEYID, NKEYID, is NonceA | NonceB

sNKEYID = hash(sKEYID, Na, Nb)

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

Each party must validate the hash values before changing any state information associated with keys.

2.8 Distribution of an External Key

Once an OAKLEY session key and anciliary algorithms are established, the keying material and the encryption 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. Only the first two fields of each message are plaintext, the rest is encrypted using the encryption algorithm associated with the KEYID state.

```
  Initiator                                                    Responder
  ---------                                                    ---------
    ->       KEYID, INEWEXTKEY,  Na, sNEWKEYID                    ->
   <-        KEYID, INEWEXTKEYRQ, Nb, hash(Nb, Na, sNEWKEYID)     <-
    ->       KEYID, INEWEXTKEYRS, hash(Na, Nb, sNEWKEYID)         ->
```

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

The new key identifier, naming the keying material sNEWKEYID, is hash(1 | Na | Nb).

2.7.1 Retransmissions, Timeouts, and Error Messages

TBD

2.8.2 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.

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. 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).

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: .sp. nf 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 repres
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
multi-precision integers.

<optional>
Largest prime factor of the group order:

380599384721589301615546382619538626639743644 3,

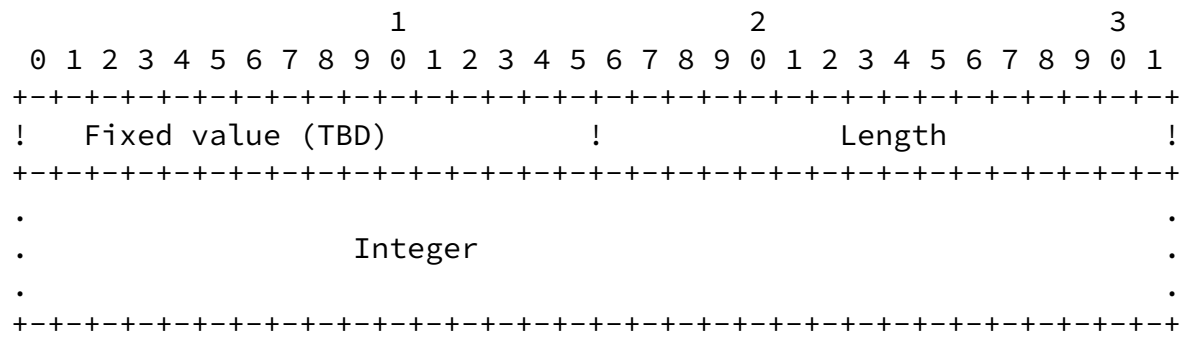
represented as a multi-precision integer.
The order of the group:

4567192616659071619386556591434463519676923731 6

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.



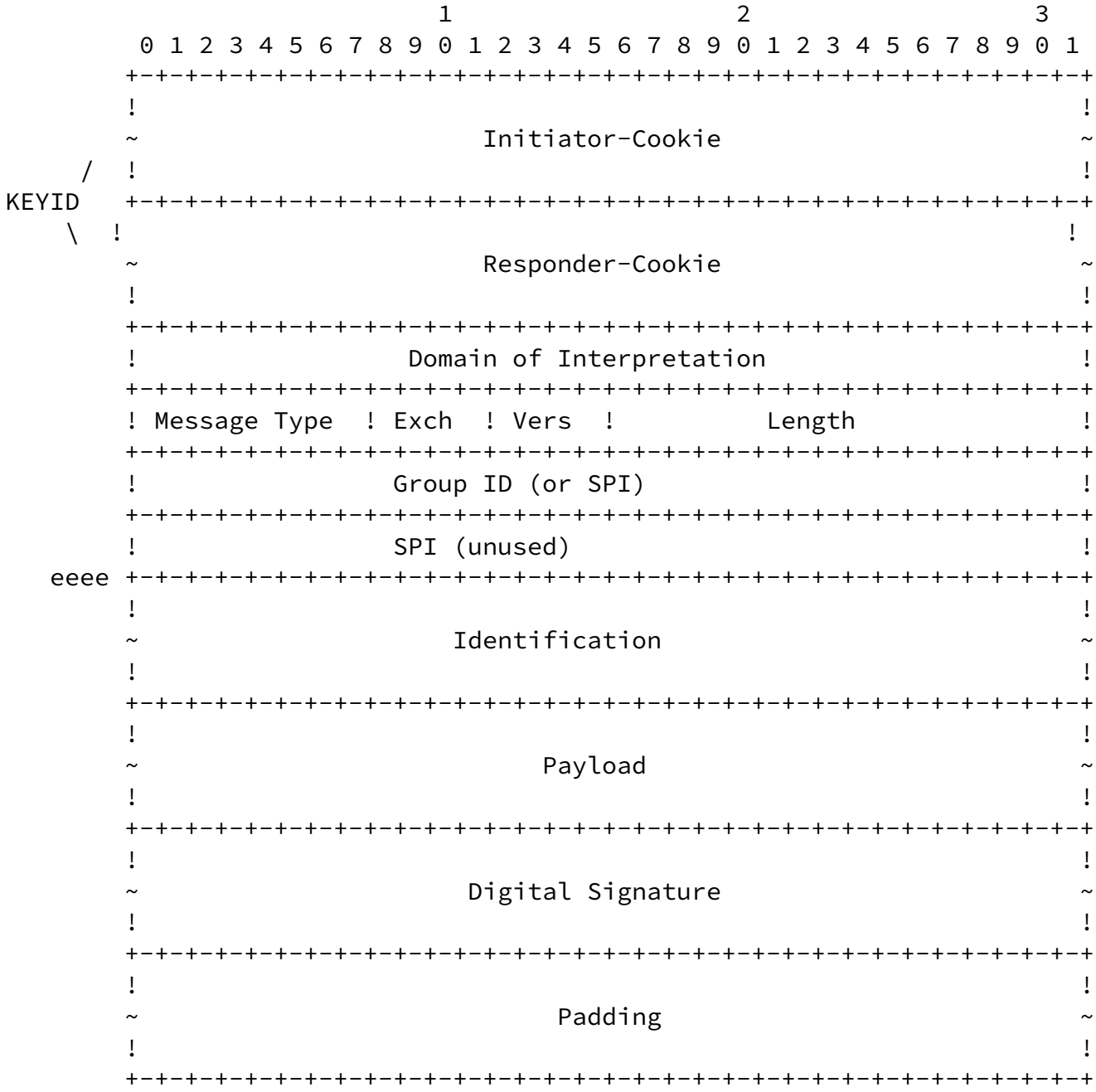
APPENDIX B New Group Definition

TBD

APPENDIX C Message format

1. Message format template

The following message format is meant to be compatible with ISAKMP formats. Any anomalies will be resolved by ongoing coordination activities.



"eeee" represents the encryption boundary for messages requiring privacy.

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.

2. Message Types

The following indicates the constant values for message types. These will be assigned unique values, although the values are TBD at the time of this writing.

```
    IREQ
    IKREQ
    IKREP
    IAUTH_REQ
    IAUTH_REP
    IAUTH_PRF
    IAUTH_PRF_R
    INEWGRP
    INEWGRRS
    INEWGRPACK
    INEWKRQ
    INEWKRS
    INEWKRP
    INEWEXTKEY
    INEWEXTKEYRQ
    INEWEXTKEYRS
```

Related ISAKMP types

```
   ISA_INIT_REQ
   ISA_INIT_RESP
   ISA_AUTH&KE_REQ
   ISA_AUTH&KE_RESP
   ISA_NEW_GROUP_REQ  (recommended addition)
   ISA_NEW_GROUP_RESP (recommended addition)
```

3. Payload

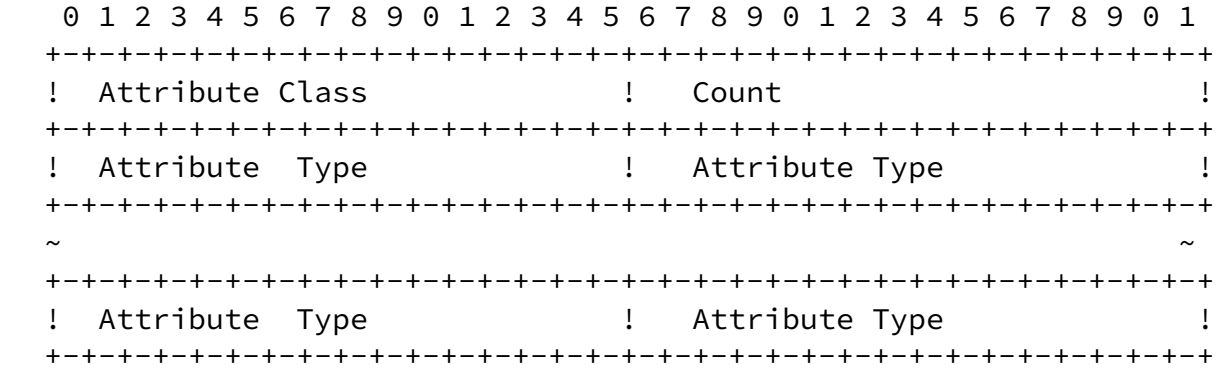
The Payload section will carry the g^x values, encoded as variable precision integers.

3.1 Generic List Exchange Format for Encryption/Hash/Authentication

Up to three attribute classes, each followed by a count and a list of algorithms. The encoding is as in ISAKMP:

a list of pairs, each one indicating its mode of use
(encryption or hashing), and the algorithm type.
The length of the list is indicated by the count field.

1 2 3



4. Identification

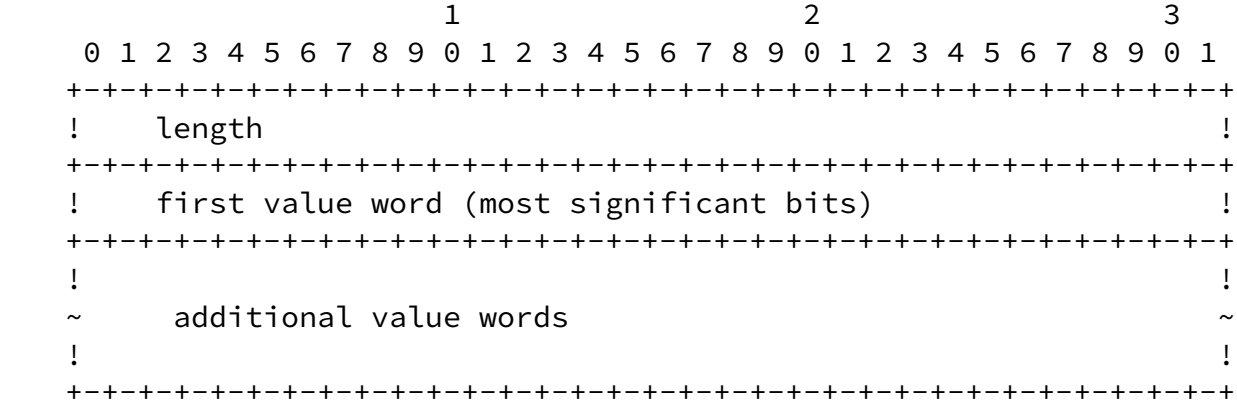
The Identification section will carry the values indicated by "ID" in the text of this document.

5. Digital Signature

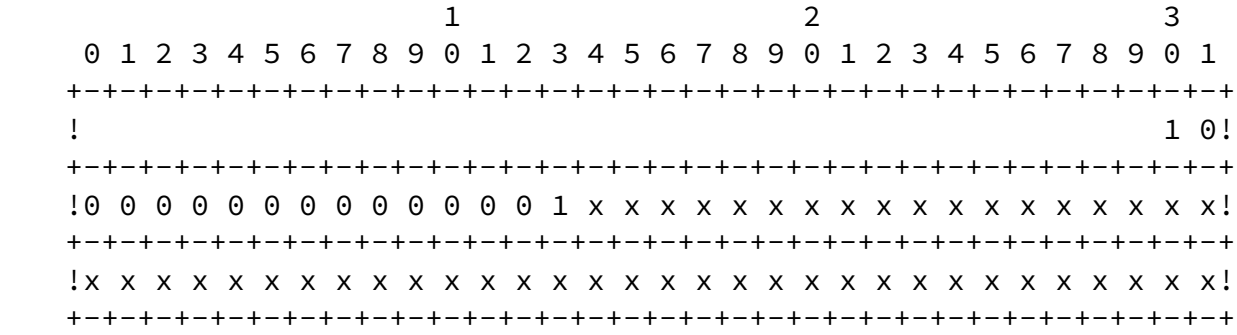
The Digital Signature section will carry the values indicated by "hash" in the text of this document.

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.



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.



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 PGP Keys for Authentication

TBD

APPENDIX G X509 Certificates

TBD

APPENDIX H DSS Certificates

The format and validation methods will be specified in an Internet draft, draft-cylink-dss-cert-00.txt.

APPENDIX I 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 KEYID's) will also be given here.

```
0   Reserved
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]

2^32 and higher are used for private group identifiers



Until then, TBD.
```

Appendix J Domain of Interpretation

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-cylink-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,

[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.