Российская Академия Наук
Институт системного программирования

Internet Key Exchange (IKEv2) Protocol

Протокол обмена ключами версии 2 (IKEv2)

RFC 4306 “Internet Key Exchange (IKEv2) Protocol”
Перевод на русский язык.

Москва 2007 г.

Список исполнителей

Исполнитель	Контактная информация	Примечания
Шнитман Виктор Зиновьевич, д.т.н., профессор	vzs@ispras.ru	Перевод.

История документа

№ версии	Дата	Примечания
1.0	01.12.2007	Перевод опубликован.

Аннотация.

В данном документе представлен перевод на русский язык RFC 4306, регламентирующего реализацию автоматического установления контекстов безопасности для защищенного обмена данными в рамках архитектуры IPsec.

Перевод выполнен в рамках проекта по гранту Российского фонда фундаментальных исследований № 07-07-00243 «Верификация функций безопасности протокола нового поколения IPsec v2».

Network Working Group
Request for Comments: 4306
Obsoletes: 2407, 2408, 2409
Category: Standards Track
C. Kaufman, Ed.
Microsoft
December 2005

Internet Key Exchange (IKEv2) Protocol

Протокол обмена ключами версии 2 (IKEv2)

Статус данного меморандума

Данный документ определяет для сообщества Internet протокол, находящийся в процессе стандартизации, и требует обсуждения, а также предложений по улучшению. За информацией, касающейся состояния стандартизации и статуса данного протокола, обращайтесь, пожалуйста, к текущей версии документа "Официальные стандарты протоколов Internet" (STD 1). Распространение данного меморандума не ограничено.

Аннотация

В данном документе описывается версия 2 протокола обмена ключами в Internet (IKE - Internet Key Exchange). IKE является частью IPsec, используемой для выполнения взаимной аутентификации, установления и поддержки контекстов безопасности (SA - security association).

Данная версия спецификации IKE объединяет содержимое того, что раньше было отдельными документами, в том числе протокол управления контекстами безопасности и ключами в Internet (ISAKMP, RFC 2408), протокол IKE (RFC 2409), домен интерпретации Internet (DOI, RFC 2407), а также вопросы пересечения устройств трансляции адресов (NAT - Network Address Translation), традиционной аутентификации и приобретения удаленного адреса.

Версия 2 IKE не обеспечивает взаимодействия с версией 1, но она имеет достаточно общий формат заголовка так, что обе версии могут однозначно работать через один и тот же UDP-порт.

Содержание

1. Введение

Протокол безопасности IP (IPsec) обеспечивает конфиденциальность, целостность данных, управление доступом и аутентификацию источника данных для IP-дейтаграмм. Эти сервисы обеспечиваются путем поддержки между источником и приемником IP-дейтаграмм общего состояния. Это состояние, помимо прочего, определяет конкретные сервисы, предоставляемые дейтаграмме, какие криптографические алгоритмы будут использоваться для обеспечения этих сервисов, а также ключи, используемые в качестве входных данных для этих криптографических алгоритмов.

Установление этого общего состояния ручными методами не очень хорошо масштабируется. Поэтому требуется протокол для динамического установления этого состояния. В данном меморандуме описывается такой протокол – протокол обмена ключами в Internet (IKE - Internet Key Exchange). Это вторая версия протокола IKE. Версия 1 IKE была определена в документах RFC 2407, 2408 и 2409 [Pip98, MSST98, HC98]. Данный единственный документ предназначен для замены всех трех указанных документов RFC.

Определения исходных терминов данного документа (например, контекста безопасности – Security Association или SA) можно найти в [RFC4301].

Ключевые слова "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" и "MAY", которые появляются в данном документе, должны интерпретироваться так, как описано в [Bra97].

Термин "Expert Review" должен интерпретироваться так, как определено в [RFC2434].

Протокол IKE выполняет взаимную аутентификацию между двумя сторонами и устанавливает контекст безопасности IKE SA, включающий общую секретную информацию, которая может использоваться для эффективного установления контекстов безопасности для протокола ESP [RFC4303] и/или протокола AH [RFC4302], а также набор криптографических алгоритмов, которые должны использоваться контекстами безопасности для защиты трафика, который передается под их защитой. В данном документе термин "набор" (suite) или "криптографический набор" (cryptographic suite) относится к полному набору алгоритмов, используемых для защиты контекста безопасности. Инициатор предлагает один или несколько наборов путем перечисления поддерживаемых алгоритмов, которые могут объединяться в наборы путем составления соответствующих комбинаций. Протокол IKE может также согласовать использование IPComp [IPCOMP] совместно с ESP SA и/или AH SA. Контекст безопасности IKE SA мы называем "IKE_SA". Контексты безопасности ESP и/или AH, которые устанавливаются посредством IKE_SA, мы называем дочерними контекстами безопасности ("CHILD_SA").

Все обмены информацией IKE состоят из пар сообщений: запроса и ответа. Пара сообщений называется "обменом" (exchange). Мы называем первые сообщения, устанавливающие IKE_SA обменами IKE_SA_INIT и IKE_AUTH, а последующие обмены IKE обменом создания дочернего контекста безопасности (CREATE_CHILD_SA) или информационным обменом (INFORMATIONAL exchange). Обычно для установления IKE_SA и первого CHILD_SA используется один обмен IKE_SA_INIT и один обмен IKE_AUTH (всего четыре сообщения). В исключительных случаях может быть более одного обмена каждого указанного типа. Во всех случаях все обмены IKE_SA_INIT должны (MUST) завершиться до начала любого обмена другого типа, затем должны (MUST) завершиться все обмены IKE_AUTH, а вслед за ними может выполняться любое количество обменов типа CREATE_CHILD_SA и INFORMATIONAL в любом порядке. В некоторых случаях между двумя оконечными точками IPsec требуется только один контекст безопасности CHILD_SA и поэтому дополнительных обменов не будет. Последующие обмены могут (MAY) использоваться для установления дополнительных дочерних контекстов безопасности CHILD_SA между той же самой аутентифицированной парой оконечных точек, а также для выполнения вспомогательных функций.

Последовательность сообщений IKE всегда включает запрос, за которым следует ответ. Обеспечение надежности является ответственностью инициатора запроса. Если ответ не получен в течение интервала тайм-аута, инициатор запроса должен повторно передать запрос (или отказаться от соединения).

В первой паре сообщений запрос/ответ сеанса IKE (IKE_SA_INIT) согласуются параметры безопасности для IKE_SA, посылаются одноразовые номера и посылаются значения Диффи-Хеллмана.

Во второй паре сообщений запрос/ответ (IKE_AUTH) пересылаются идентификаторы партнеров, подтверждается знание секретов, соответствующих обоим идентификаторам, и устанавливается контекст безопасности (SA) для первого (и часто единственного) AH и/или ESP CHILD_SA.

Последующие обмены могут быть типа CREATE_CHILD_SA (который создает CHILD_SA) и INFORMATIONAL (который удаляет SA, сообщает условия ошибки или выполняет другие вспомогательные функции). Каждый запрос требует ответа. Запрос типа INFORMATIONAL без блоков данных (отличных от пустого блока данных Шифр, требуемого синтаксисом) обычно используется в качестве проверки работоспособности (liveness) партнера. Эти последующие обмены не могут использоваться до тех пор, пока не закончатся начальные обмены.

В нижеследующем описании мы предполагаем, что ошибок не происходит. Изменения к ходу обмена в случае возникновения ошибок описаны в подразделе 2.21.

1.1. Сценарии использования

Предполагается, что IKE должен использоваться для согласования контекстов безопасности ESP и/или AH в различных сценариях, каждый из которых имеет свои собственные специальные требования.

1.1.1. Туннель между защитными шлюзами

```
                    +-+-+-+-+-+            +-+-+-+-+-+
                    !         ! Туннель    !         !
      Защищенная    !Оконечная! IPsec      !Оконечная!     Защищенная
      подсеть   <-->!точка    !<---------->!точка    !<--> подсеть
                    !туннеля  !            !туннеля  !
                    +-+-+-+-+-+            +-+-+-+-+-+
```

Рис. 1. Туннель между защитными шлюзами.

В этом сценарии ни одна из оконечных точек IP-соединения не реализует IPsec, но узлы сети, находящиеся между ними, защищают трафик на части пути. Для оконечных точек защита прозрачна и зависит от обычной маршрутизации, обеспечивающей посылку пакетов для обработки через оконечные точки туннеля. Каждая оконечная точка объявит набор адресов, находящихся "за ней", и пакеты будут посылаться в туннельном режиме, при котором внутренний IP-заголовок будет содержать IP-адреса реальных оконечных точек.

1.1.2. Транспортировка между оконечными точками

```
       +-+-+-+-+-+-+                                           +-+-+-+-+-+-+
       !           !     Контекст безопасности IPsec           !           !
       !Защищенная !     в транспортном или туннельном режиме !Защищенная !
       !оконечная  !<----------------------------------------->!оконечная  !
       !точка      !                                           !точка      !
       +-+-+-+-+-+-+                                           +-+-+-+-+-+-+
```

Рис. 2. Транспортировка между оконечными точками.

В этом сценарии обе оконечные точки IP-соединения реализуют IPsec, как это требуется от хостов в [RFC4301]. Обычно будет использоваться транспортный режим без внутреннего IP-заголовка. Если имеется внутренний IP-заголовок, то внутренние адреса будут совпадать с внешними адресами. Для пакетов, которые должны быть защищены этим SA, будет согласовываться одна пара адресов. Базируясь на аутентифицированных IPsec идентификаторах участников, эти оконечные точки могут (MAY) реализовать управление доступом прикладного уровня. Этот сценарий дает возможность организовать сквозную защиту, которая считалась, начиная с [RFC1958], [RFC2775], руководящим принципом Internet и методом ограничения неустранимых проблем со сложностью в сетях, указанном в [RFC3439]. Хотя этот сценарий не может быть полностью применен к IPv4 Internet, он успешно применялся в конкретных ситуациях в рамках интранет-сетей, использующих IKEv1. Он должен получить широкое распространение в процессе перехода на IPv6 и внедрения IKEv2.

В этом сценарии допускается, чтобы одна или обе защищенные оконечные точки находились за узлом, выполняющим трансляцию сетевых адресов (NAT - network address translation). В этом случае туннелируемые пакеты должны быть инкапсулированы в UDP так, чтобы номера портов в UDP-заголовках могли использоваться для указания отдельных оконечных точек, "находящихся за" NAT (см. подраздел 2.23).

1.1.3. Транспортировка между оконечной точкой и защитным шлюзом

```
+-+-+-+-+-+-+                          +-+-+-+-+-+
!           !                          !         !     Защищенная
!Защищенная !   Туннель IPsec          !Оконечная!     подсеть
!оконечная  !&lt;------------------------&gt;!точка    !&lt;--- и/или
!точка      !                          !туннеля  !     Internet
+-+-+-+-+-+-+                          +-+-+-+-+-+
```

Рис. 3. Транспортировка между оконечной точкой и защитным шлюзом.

В этом сценарии защищенная оконечная точка (обычно портативный перемещающийся компьютер) соединяется со своей корпоративной сетью через защищенный IPsec-туннель. Она может использовать этот туннель только для доступа к информации в корпоративной сети, либо она может туннелировать весь свой трафик через корпоративную сеть, чтобы воспользоваться преимуществами защиты от атак со стороны Internet, обеспечиваемой корпоративным межсетевым экраном. В любом случае защищенная оконечная точка захочет, чтобы с защитным шлюзом был ассоциирован IP-адрес таким образом, чтобы пакеты, возвращаемые на этот адрес, поступали на защитный шлюз и туннелировались назад. Этот IP-адрес может быть статическим или может динамически распределяться защитным шлюзом. Для поддержки последнего случая IKEv2 включает механизм запроса инициатором IP-адреса, владельцем которого является защитный шлюз, который может использоваться в течение времени жизни его контекста безопасности (SA).

В этом сценарии пакеты будут использовать туннельный режим. В каждом пакете, поступающем из защищенной оконечной точки, внешний IP-заголовок будет содержать IP-адрес источника, связанный с его текущим местоположением (т.е. адрес, на который будет поступать трафик, маршрутизируемый непосредственно на оконечную точку), в то время как внутренний IP-заголовок будет содержать IP-адрес источника, присвоенный защитным шлюзом (т.е. адрес, на который будет поступать трафик, маршрутизируемый на защитный шлюз для пересылки оконечной точке). Внешний адрес места назначения всегда будет адресом защитного шлюза, в то время как внутренний адрес места назначения будет конечным местом назначения пакета.

В этом сценарии, возможно, защищенная оконечная точка будет находиться за NAT. В этом случае IP-адрес, видимый защитным шлюзом, не будет совпадать с IP-адресом, посылаемым защищенной оконечной точкой, и пакеты должны быть инкапсулированы в UDP, чтобы обеспечить правильную маршрутизацию.

1.1.4. Другие сценарии

Возможны и другие сценарии, поскольку они являются вложенными комбинациями рассмотренных выше сценариев. В известном примере комбинируются аспекты пунктов 1.1.1 и 1.1.3. Некоторая подсеть может выполнять все внешние обращения через удаленный защитный шлюз, используя IPsec-туннель, когда оставшаяся часть Internet маршрутизирует адреса подсети на защитный шлюз. В качестве примера может служить чья-то домашняя сеть, находящаяся фактически в Internet со статическими IP-адресами, хотя коннективность обеспечивается поставщиком услуг Internet (ISP – Internet Service Provider), который присваивает один динамически назначенный адрес защитному шлюзу пользователя (когда статические IP-адреса и IPsec-ретранслятор предоставляются третьей стороной, располагающейся где угодно).

1.2. Начальные обмены

Обмен информацией, используемый IKE, всегда начинается с обменов IKE_SA_INIT и IKE_AUTH (называемых в IKEv1 фазой 1). Эти начальные обмены обычно включают четыре сообщения, хотя в некоторых сценариях их количество может увеличиться. Все обмены информацией, используемые IKE, состоят из пар сообщений запрос/ответ. Сначала мы опишем основной обмен, за которым последуют варианты. Первая пара сообщений (IKE_SA_INIT) согласует криптографические алгоритмы, осуществляет обмен одноразовыми номерами и выполняет обмен Диффи-Хеллмана.

Вторая пара сообщений (IKE_AUTH) аутентифицирует предыдущие сообщения, осуществляет обмен идентификаторами и сертификатами, а также устанавливает первый дочерний контекст безопасности (CHILD_SA). Части этих сообщений зашифровываются и защищаются для обеспечения целостности с помощью ключей, установленных в процессе обмена IKE_SA_INIT таким образом, что идентификаторы оказываются скрытыми от подслушивающих, и все поля во всех сообщениях оказываются аутентифицированными.

В последующем описании блоки данных, содержащиеся в сообщении, указываются именами, которые перечислены ниже.

Обозначение	Блок данных
AUTH	Authentication (Аутентификация)
CERT	Certificate (Сертификат)
CERTREQ	Certificate Request (Запрос Сертификата)
CP	Configuration (Конфигурирование)
D	Delete (Удаление)
E	Encrypted (Шифр, зашифрованный блок данных)
EAP	Extensible Authentication (Расширяемая Аутентификация)
HDR	IKE Header (Заголовок IKE)
IDi	Identification – Initiator (Идентификация – инициатор)
IDr	Identification – Responder (Идентификация - ответчик)
KE	Key Exchange (Обмен Ключами)
Ni, Nr	Nonce (Одноразовый Номер)
N	Notify (Уведомление)
SA	Security Association (Контекст Безопасности)
TSi	Traffic Selector – Initiator (Селекторы Трафика – инициатор)
TSr	Traffic Selector – Responder (Селекторы Трафика – ответчик)
V	Vendor ID (Идентификатор Поставщика)

Детали содержимого блока данных каждого типа описываются в разделе 3. Необязательные (факультативные) блоки данных, которые могут появляться в сообщении, будут заключаться в скобки, например, запись [CERTREQ], указывает на то, что в сообщение дополнительно может включаться блок данных Запрос Сертификата.

Начальные обмены выглядят следующим образом:

```
       Инициатор                          Ответчик
      -----------                        -----------
       HDR, SAi1, KEi, Ni   -->
```

Заголовок HDR содержит индексы параметров безопасности (SPI – Security Parameter Index), номера версий и различного вида флаги. Блок данных SAi1 определяет криптографические алгоритмы, которые поддерживает инициатор для IKE_SA. Блок данных KEi содержит значение Диффи-Хеллмана инициатора. Блок данных Ni – это одноразовый номер инициатора.

```
                                 <--    HDR, SAr1, KEr, Nr, [CERTREQ]
```

Ответчик выбирает криптографический набор из предложенных на выбор инициатором и отправляет этот выбор в блоке данных SAr1, завершает обмен Диффи-Хеллмана с помощью блока данных KEr и посылает свой одноразовый номер в блоке данных Nr.

В этой точке согласования каждая из сторон может сгенерировать порождающий ключ SKEYSEED, из которого формируются все ключи для данного IKE_SA. Все, кроме заголовков всех последующих сообщений, зашифровывается, и защищается целостность передаваемых данных. Ключи, используемые для шифрования и защиты целостности, формируются из SKEYSEED и называются ключом шифрования SK_e (от encryption) и ключом аутентификации SK_a (от authentication, что, по сути, означает защиту целостности). Для каждого направления вычисляются отдельные ключи SK_e и SK_a. Кроме ключей SK_e и SK_a, формируемых на основе значения Диффи-Хеллмана для защиты контекста безопасности IKE_SA, вычисляется величина SK_d, которая используется для формирования дополнительного ключевого материала для дочерних контекстов безопасности (CHILD_SA). Запись SK { ... } указывает на то, что эти данные зашифровываются и защищается их целостность с помощью ключей SK_e и SK_a соответствующего направления.

```
       HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,]
                  AUTH, SAi2, TSi, TSr}     -->
```

Инициатор объявляет свой идентификатор с помощью блока данных IDi, доказывает знание секрета, соответствующего IDi, и защищает целостность содержимого первого сообщения с помощью блока данных AUTH (см. подраздел 2.15). Он может также послать свой сертификат (свои сертификаты) в блоке данных (блоках данных) CERT и список своих доверительных анкеров в блоке данных (блоках данных) CERTREQ. Если включаются какие-либо блоки данных CERT, то первый предоставляемый сертификат должен (MUST) содержать открытый ключ, используемый для проверки поля AUTH. Факультативные блоки данных IDr позволяют инициатору установить, с каким конкретно идентификатором ответчика он хочет разговаривать. Это полезно, когда машина, на которой работает ответчик, обеспечивает хостинг нескольких идентификаторов на одном и том же IP-адресе. Инициатор начинает согласование CHILD_SA с помощью блока данных SAi2. Последние поля (начиная с SAi2) описываются в описании обмена CREATE_CHILD_SA.

```
                                        <--    HDR, SK {IDr, [CERT,] AUTH,
                                                     SAr2, TSi, TSr}
```

Ответчик объявляет свой идентификатор с помощью блока данных IDr, факультативно посылает один или несколько сертификатов (снова с первым сертификатом, содержащим открытый ключ, используемый для проверки AUTH), аутентифицирует свой идентификатор и защищает целостность второго сообщения блоком данных AUTH, а также завершает согласование CHILD_SA дополнительными полями, описанными ниже в обмене CREATE_CHILD_SA.

Получатели сообщений 3 и 4 должны (MUST) проверить, что все подписи и коды аутентификации сообщений (MAC – Message Authentication Code) вычисляются правильно и что имена в блоках данных ID соответствуют ключам, которые использовались для формирования блока данных AUTH.

1.3. Обмен CREATE_CHILD_SA

Этот обмен состоит из одной пары сообщений запрос/ответ, и в IKEv1 назывался обменом фазы 2. Он может (MAY) инициироваться любой стороной IKE_SA после завершения начальных обменов.

Все сообщения, следующие за начальным обменом, криптографически защищаются с помощью криптографических алгоритмов и ключей, согласованных в первых двух сообщениях обмена IKE. Эти последующие сообщения используют синтаксис блока данных Шифр (Encrypted Payload), описанный в подразделе 3.14. Все последующие сообщения включают блок данных Шифр, даже если в данном тексте они указаны как "пустые".

Любая оконечная точка может инициировать обмен CREATE_CHILD_SA, поэтому в данном подразделе термин инициатор относится к той оконечной точке, которая инициировала данный обмен.

Контекст безопасности CHILD_SA создается путем посылки запроса CREATE_CHILD_SA. Запрос CREATE_CHILD_SA может (MAY) в качестве опции содержать блок данных KE для дополнительного обмена Диффи-Хеллмана, чтобы обеспечить более сильные гарантии прогрессирующей секретности для CHILD_SA. Ключевой материал для CHILD_SA является функцией ключа SK_d, установленного в процессе установления IKE_SA, одноразовых номеров, обмен которыми произошел в процессе обмена CREATE_CHILD_SA, и значения Диффи-Хеллмана (если в обмен CREATE_CHILD_SA включаются блоки данных KE).

В контексте безопасности CHILD_SA, созданном как часть начального обмена, второй блок данных KE и одноразовый номер не должны (MUST NOT) посылаться. При вычислении ключей для CHILD_SA используются одноразовые номера из начального обмена.

Запрос CREATE_CHILD_SA содержит:

```
       Инициатор                                 Ответчик
      -----------                               -----------
       HDR, SK {[N], SA, Ni, [KEi],
           [TSi, TSr]}             -->
```

Инициатор посылает предложение (предложения) SA в блоке данных SA, одноразовый номер в блоке данных Ni, факультативно значение Диффи-Хеллмана в блоке данных KEi и предлагаемые селекторы трафика в блоках данных TSi и TSr. Если этот обмен CREATE_CHILD_SA представляет собой переустановку (rekeying) существующего SA, отличного от IKE_SA, то находящийся впереди блок данных N типа REKEY_SA должен (MUST) идентифицировать переустанавливаемый SA. Если этот обмен CREATE_CHILD_SA не является переустановкой существующего контекста безопасности (SA), то блок данных N должен (MUST) быть опущен. Если предложения SA содержат различные группы Диффи-Хеллмана, то KEi должен (MUST) быть элементом группы, которую по предположению инициатора примет ответчик. Если его предположения окажутся не верными, то обмен CREATE_CHILD_SA закончится неудачей, и он должен будет снова повторить попытку с другим KEi.

Сообщение, следующее за заголовком, зашифровывается, и целостность сообщения, включая заголовок, защищается с помощью криптографического алгоритма, согласованного для IKE_SA.

Ответ CREATE_CHILD_SA включает:

```
                                  <--    HDR, SK {SA, Nr, [KEr],
                                               [TSi, TSr]}
```

Ответчик отвечает (используя для ответа тот же самый идентификатор сообщения - Message ID) принятым предложением в блоке данных SA и значением Диффи-Хеллмана в блоке данных KEr, если блок данных KEi был включен в запрос и выбранный криптографический набор включает эту группу. Если ответчик выбирает криптографический набор с другой группой, то он должен (MUST) отклонить запрос. Инициатор должен (SHOULD) повторить запрос, но теперь с блоком данных KEi из группы, выбранной ответчиком.

Селекторы трафика для трафика, который должен посылаться под защитой данного контекста безопасности (SA), определяются в блоках данных TS, которые могут представлять собой подмножество того, что предлагал инициатор CHILD_SA. Селекторы трафика опускаются, если этот запрос CREATE_CHILD_SA используется для изменения ключа IKE_SA.

1.4. Информационный обмен

В различные моменты времени в процессе работы IKE_SA партнеры могут захотеть передать друг другу управляющие сообщения, касающиеся ошибок или уведомлений об определенных событиях. Чтобы это выполнить, IKE определяет информационный обмен (INFORMATIONAL exchange). Информационные обмены должны (MUST) выполняться только (ONLY) после начальных обменов, и криптографически защищаются согласованными ключами.

Управляющие сообщения, которые относятся к некоторому IKE_SA, должны (MUST) посылаться под защитой этого IKE_SA. Управляющие сообщения, которые относятся к контекстам безопасности CHILD_SA, должны (MUST) посылаться под защитой того IKE_SA, который их генерировал (или его приемника, если IKE_SA был заменен с целью переустановки).

Сообщения в информационном обмене содержат ноль или несколько блоков данных Уведомление (Notification), Удаление (Delete) и Конфигурирование (Configuration). Получатель информационного сообщения запроса должен (MUST) послать некоторый ответ (иначе отправитель предположит, что сообщение было потеряно в сети, и будет его передавать повторно). Этот ответ может (MAY) быть сообщением без каких-либо блоков данных. Сообщение запроса в информационном обмене также может (MAY) не содержать никаких блоков данных. Это предполагаемый способ, с помощью которого оконечная точка может попросить другую оконечную точку проверить, что она жива.

Контексты безопасности ESP SA и AH SA всегда существуют парами, по одному контексту безопасности в каждом направлении. Когда закрывается контекст безопасности, оба элемента пары должны (MUST) быть закрыты. Если контексты безопасности между одной и той же парой оконечных точек являются вложенными, например, в случае, когда данные (и IP-заголовки в туннельном режиме) инкапсулируются сначала протоколом IPComp, затем протоколом ESP, и, наконец, протоколом AH, все контексты безопасности должны (MUST) уничтожаться вместе. Каждая оконечная точка должна (MUST) закрыть свои входящие контексты безопасности и позволить другой оконечной точке закрыть другой контекст безопасности в каждой паре. Для уничтожения контекста безопасности посылается информационный обмен с одним или несколькими блоками данных Удаление, в которых перечисляются индексы параметров безопасности (SPIs) для контекстов безопасности, которые подлежат удалению (поскольку они предполагаются в заголовках входящих пакетов). Получатель должен (MUST) закрыть указанные контексты безопасности. Обычно ответ в информационном обмене будет содержать блоки данных Удаление для парных контекстов безопасности, идущих в обратном направлении. Имеется одно исключение. Если случайно обе оконечные точки множества контекстов безопасности независимо решат их закрыть, то каждая из них может послать блок данных Удаление, и оба запроса могут пересечься в сети. Если узел получает запрос на удаление контекстов безопасности, для которых он уже выдал запрос удаления, он должен (MUST) удалить исходящие контексты безопасности в процессе обработки запроса и входящие контексты безопасности в процессе обработки ответа. В этом случае ответы не должны (MUST NOT) включать блоки данных Удаление для удаляемых контекстов безопасности, поскольку это приведет к дублированию удаления и теоретически может удалить неверный контекст безопасности.

Узел должен (SHOULD) рассматривать наполовину закрытые соединения как аномальные и проверять, продолжают ли они существовать. Заметим, что данная спецификация нигде не определяет периоды времени, так что решение о продолжительности ожидания оставлено на усмотрение отдельных оконечных точек. Узел может (MAY) отказаться принимать входящие данные по наполовину закрытым соединениям, но не должен (MUST NOT) в одностороннем порядке их закрывать и повторно использовать индексы параметров безопасности. Если состояние соединения становится совершенно испорченным, узел может (MAY) закрыть IKE_SA, который неявно закроет все контексты безопасности, согласованные под его защитой. Затем он может заново создать контексты безопасности, в которых он нуждается, на чистой основе под защитой нового IKE_SA.

Информационный обмен определяется следующим образом:

```
       Инициатор                            Ответчик
      -----------                          -----------
       HDR, SK {[N,] [D,] [CP,] ...} -->
                                      <-- HDR, SK {[N,] [D,] [CP], ...}
```

Обработка информационного обмена определяется составом его блоков данных.

1.5. Информационные сообщения за рамками IKE_SA

Если зашифрованный IKE-пакет поступает на порт 500 или 4500 с неопознанным индексом параметров безопасности (SPI), то это может произойти из-за того, что принимающий узел недавно потерпел аварию и потерял состояние, или по причине какой-то другой неправильной работы системы или атаки. Если принимающий узел имеет активный IKE_SA на IP-адрес, с которого пришел пакет, он может (MAY) послать в информационном обмене уведомление о неуправляемом пакете через этот IKE_SA. Если он не имеет такого IKE_SA, он может (MAY) послать информационное сообщение без криптографической защиты на IP-адрес источника. Такое сообщение не является частью информационного обмена, и принимающий узел не должен (MUST NOT) на него отвечать. Если он это сделает, может возникнуть зацикливание сообщений.

2. Детали и варианты протокола IKE

Обычно IKE осуществляет прослушивание и посылку сообщений на порт 500 UDP, хотя сообщения IKE могут также приниматься на порт 4500 UDP в немного измененном формате (см. подраздел 2.23). Поскольку UDP является дейтаграммным (ненадежным) протоколом, IKE включает в свое определение средства восстановления от ошибок передачи, в том числе от потерь пакетов, повторного воспроизведения пакетов и подделки пакетов. IKE разрабатывается для работы при условии, что (1) по крайней мере один из серии повторно передаваемых пакетов достигнет своего места назначения до истечения тайм-аута; и (2) канал не настолько заполнен подделанными и повторно воспроизводимыми пакетами, чтобы исчерпать возможности сети или процессора любой оконечной точки. Даже при отсутствии этих минимальных требований к производительности IKE разработан так, чтобы отказ происходил аккуратно (как если бы была поломана сеть).

Хотя предполагается, что сообщения IKEv2 являются короткими, они содержат структуры данных, не имеющие жесткой верхней границы размера (в частности, сертификаты X.509), а сам IKEv2 не имеет механизма для фрагментации больших сообщений. IP определяет механизм для фрагментации UDP-сообщений большого размера, но реализации различаются по поддерживаемому максимальному размеру сообщений. Более того, использование IP-фрагментации открывает реализации для атак на доступность (DoS attacks) [KPS03]. Наконец, некоторые реализации NAT и/или межсетевых экранов могут блокировать IP-фрагментацию.

Все реализации IKEv2 должны (MUST) быть способными осуществлять посылку, прием и обработку сообщения IKE размером до 1280 байт, и они должны (SHOULD) быть способными осуществлять посылку, прием и обработку сообщения размером до 3000 байт. Реализации IKEv2 должны (SHOULD) быть осведомлены о максимальном поддерживаемом размере UDP-сообщений, и они могут (MAY) усекать сообщения путем опускания каких-то сертификатов или предложений криптографических наборов, если это позволит сделать размер сообщения меньше максимального. Использование, где это возможно, форматов "Hash and URL" вместо включения в обмены сертификатов, может позволить обойти все проблемы. Однако реализации и конфигурирование должны иметь в виду, что если поиск универсальных локаторов ресурсов (URL) возможен только после установления IPsec SA, то проблемы рекурсии могут помешать этому методу работать.

2.1. Использование таймеров повторной передачи

Все сообщения в IKE существуют парами: запрос и ответ. Установка IKE_SA обычно состоит из двух пар запрос/ответ. После установки IKE_SA любой конец контекста безопасности может инициировать запросы в любое время, и в любой данный момент времени "в полете" может находиться много запросов и ответов. Но каждое сообщение помечается признаком запроса или ответа, и для каждой пары запрос/ответ один из концов контекста безопасности является инициатором, а другой – ответчиком.

Для каждой пары сообщений IKE ответственность за повторные передачи в случае тайм-аута несет инициатор. Ответчик никогда не должен (MUST) повторно передавать ответ, кроме случая, когда он принимает повторную передачу запроса. В этом случае ответчик должен (MUST) полностью игнорировать повторно переданный запрос за исключением инициации повторной передачи ответа. Инициатор должен (MUST) хранить каждый запрос до тех пор, пока он не получит соответствующего ответа. Ответчик должен (MUST) хранить каждый ответ до тех пор, пока он не получит запрос, порядковый номер которого больше, чем порядковый номер в ответе плюс размер его окна (см. подраздел 2.3).

IKE является надежным протоколом в том смысле, что инициатор должен (MUST) повторно передавать запрос до тех пор, пока он либо не получит соответствующий ответ, либо не сочтет, что установление контекста безопасности IKE не удалось и он не сбросит все состояние, связанное с IKE_SA и любыми CHILD_SA, согласованными с использованием этого IKE_SA.

2.2. Использование порядковых номеров для идентификатора сообщения

Каждое сообщение IKE содержит поле Message ID (идентификатор сообщения), которое является частью его фиксированного заголовка. Этот идентификатор сообщения используется для сопоставления запросов и ответов, а также для идентификации повторных передач сообщений.

Идентификатор сообщения представляет собой 32-битовую величину, которая равна нулю для первого IKE-запроса в каждом направлении. Сообщения начальной установки IKE_SA всегда нумеруются 0 и 1. Каждая оконечная точка контекста безопасности IKE поддерживает два "текущих" идентификатора сообщений: следующий идентификатор, который должен использоваться для инициируемого им запроса, и следующий идентификатор, который он предполагает увидеть в запросе от другого конца. Эти счетчики инкрементируются по мере генерации и приема сообщений. Ответы всегда содержат тот же самый идентификатор сообщения, что и соответствующий запрос. Это означает, что после начального обмена любое целое n может появиться в качестве идентификатора сообщения в четырех различных сообщениях: n-ном запросе от первоначального инициатора IKE, соответствующем ответе, n-ном запросе от первоначального ответчика IKE и соответствующем ответе. Если два конца соединения выполняют существенно разное количество запросов, то идентификаторы сообщений в двух направлениях могут сильно отличаться. Однако неоднозначность в сообщениях отсутствует, поскольку биты (I)nitiator и (R)esponse в заголовке сообщения определяют, каким конкретным сообщением из четырех возможных является данное сообщение.

Заметим, что идентификаторы сообщений защищаются криптографически и обеспечивают защиту от повторного воспроизведения сообщений. В маловероятном случае, когда идентификаторы сообщений становятся слишком большими и "не влезают" в 32 бита, IKE_SA должен (MUST) быть закрыт. Переустановка IKE_SA сбрасывает порядковые номера.

2.3. Размер окна для перекрывающихся запросов

Для максимизации пропускной способности IKE оконечная точка IKE может (MAY) выдавать несколько запросов до получения ответа на любой из них, если другая оконечная точка указала свою способность обрабатывать такие запросы. Для простоты реализация IKE может (MAY) решить обрабатывать запросы строго упорядоченно и/или подождать ответа на один запрос до выдачи другого. Для обеспечения взаимодействия между реализациями, использующими различные стратегии, необходимо следовать определенным правилам.

После установления IKE_SA любой из концов соединения может инициировать один или несколько запросов. Эти запросы могут обгонять друг друга в сети. Чтобы избежать этой тупиковой ситуации, оконечная точка IKE должна (MUST) быть готова принять и обработать некоторый запрос, несмотря на то, что она имеет запрос, не обработанный подобающим образом. Оконечная точка IKE должна (SHOULD) быть готова принять и обработать несколько запросов, несмотря на то, что она имеет необработанный запрос.

Оконечная точка IKE должна (MUST) дожидаться ответа на каждое свое сообщение до посылки последующего сообщения до тех пор, пока она от своего партнера не получит сообщение Уведомление SET_WINDOW_SIZE, информирующее ее о том, что партнер подготовился к поддержке состояния для нескольких исходящих сообщений, чтобы обеспечить большую пропускную способность.

Оконечная точка IKE не должна (MUST NOT) превышать установленный партнером размер окна для передаваемых запросов IKE. Другими словами, если ответчик установил, что размер его окна равен N, то когда инициатору требуется выполнить запрос X, он должен (MUST) подождать до тех пор, пока он не получит ответы на все запросы вплоть до запроса X-N. Оконечная точка IKE должна (MUST) хранить копию каждого запроса, который она послала (или быть способной точно сформировать его заново) до тех пор, пока она не получит соответствующий ответ. Оконечная точка IKE должна (MUST) хранить копию количества предыдущих ответов (или быть способной точно восстановить его заново) равную своему объявленному размеру окна в случае, если ее ответ был потерян и инициатор запрашивает его повторную передачу путем повторной передачи запроса.

Оконечная точка IKE, поддерживающая размер окна больший, чем единица, должна (SHOULD) быть способной обрабатывать входящие запросы неупорядоченно для максимизации производительности в случае отказов сети или переупорядочивания пакетов.

2.4. Синхронизация состояния и тайм-ауты соединений

Оконечной точке IKE в любой момент времени позволено забывать все свое состояние, ассоциированное с IKE_SA и совокупностью соответствующих CHILD_SA. Это ожидаемое поведение в случае отказа и перезапуска оконечной точки. Важно, чтобы при отказе или повторной инициализации состояния одной оконечной точки, другая оконечная точка обнаруживала эти ситуации и не продолжала тратить впустую полосу пропускания сети путем посылки пакетов через сброшенные контексты безопасности и не давала им падать в черную дыру.

Поскольку IKE разработан для работы в условиях наличия со стороны сети атак на доступность (DoS attacks), оконечная точка не должна (MUST NOT) делать вывод о том, что в другой оконечной точке произошел отказ, базируясь на какой-либо информации о маршрутизации (например, на сообщениях ICMP) или на сообщениях IKE, которые поступают без криптографической защиты (например, на сообщениях Уведомление, жалующихся на неизвестные индексы параметров безопасности). Оконечная точка должна (MUST) делать вывод о том, что в другой оконечной точке произошел отказ, только когда повторяющиеся попытки связаться с ней остались без ответа в течение периода тайм-аута или когда получено криптографически защищенное уведомление INITIAL_CONTACT по другому IKE_SA на тот же самый аутентифицированный идентификатор. Оконечная точка должна (SHOULD) допускать, что в другой оконечной точке произошел отказ, базируясь на информации о маршрутизации, и инициировать запрос, чтобы увидеть, находится ли другая оконечная точка в работоспособном состоянии. Для проверки того, что другая сторона жива, протокол IKE определяет пустое информационное сообщение, которое (подобно всем запросам IKE) требует подтверждения (заметим, что в рамках контекста IKE_SA "пустое" сообщение состоит из заголовка IKE, за которым следует блок данных Шифр, который никаких блоков данных не содержит). Если криптографически защищенное сообщение было получено от другой стороны недавно, то незащищенные уведомления могут (MAY) игнорироваться. Реализации должны (MUST) ограничивать скорость, с которой они выполняют действия, базируясь на незащищенных сообщениях.

Количество повторных попыток и длительность тайм-аутов в данной спецификации не рассматриваются, поскольку они не связаны с интероперабельностью. Предлагается, чтобы до отказа от контекста безопасности сообщения повторно передавались не менее дюжины раз в течение периода времени, составляющего не менее нескольких минут, но в различных средах могут потребоваться различные правила. Чтобы считаться законопослушным гражданином сети, время повторной передачи должно (MUST) увеличиваться экспоненциально, что позволяет избежать затопления сети и не ухудшать существующую ситуацию с ее перегрузкой. Если на всех контекстах безопасности, ассоциированных с некоторым IKE_SA, существовал только исходящий трафик, то важно подтвердить работоспособность другой оконечной точки, чтобы избежать черных дыр. Если по IKE_SA или одному из его CHILD_SA никаких криптографически защищенных сообщений недавно не принималось, то чтобы предотвратить посылку сообщений мертвому партнеру система должна выполнить проверку его работоспособности. Получение свежего криптографически защищенного сообщения по IKE_SA или по любому его CHILD_SA гарантирует работоспособность IKE_SA и всех его CHILD_SA. Заметим, что это накладывает требования на режимы неисправности оконечной точки IKE. Реализация не должна (MUST NOT) продолжать посылку пакетов по любому контексту безопасности, если какая-либо неисправность препятствует приему по всем ассоциированным с ним контекстам безопасности. Если контексты безопасности CHILD_SA могут отказывать независимо друг от друга и ассоциированный IKE_SA не способен послать сообщение Удаление, то такие CHILD_SA должны (MUST) согласовываться посредством отдельных IKE_SA.

Существует атака на доступность инициатора IKE_SA, которую можно избежать, если инициатор проявит надлежащую осторожность. Поскольку первые два сообщения установки SA криптографически не защищены, злоумышленник может ответить на сообщение инициатора раньше истинного ответчика и поглотить попытку установления соединения. Чтобы такую атаку предотвратить, инициатор может (MAY) быть готов принять несколько ответов на свое первое сообщение, обработать каждый ответ как потенциально правильный, ответить на него, а затем отбросить все недействительные наполовину открытые соединения, когда он получит правильный криптографически защищенный ответ на любой из своих запросов. После получения криптографически правильного ответа все последующие ответы должны игнорироваться независимо от того, являются ли они криптографически правильными.

Заметим, что при таких правилах нет смысла согласовывать и договариваться относительно времени жизни SA. Если IKE предполагает, что партнер мертв, базируясь на продолжающемся отсутствии подтверждения на сообщение IKE, то тогда этот IKE SA и все CHILD_SA, установленные через этот IKE_SA, удаляются.

Оконечная точка IKE может в любой момент времени удалить неактивные CHILD_SA для восстановления ресурсов, используемых для хранения их состояния. Если оконечная точка IKE принимает решение удалить контексты безопасности CHILD_SA, она должна (MUST) послать другому концу соединения блок данных Удаление (Delete), уведомив его об удалении. Она может (MAY) подобным образом прервать IKE_SA по тайм-ауту. Закрытие IKE_SA неявно закрывает все ассоциированные с ним CHILD_SA. В этом случае, оконечная точка IKE должна (SHOULD) послать блок данных Удаление (Delete), указывая, что она закрыла IKE_SA.

2.5. Номера версий и совместимость снизу вверх

В данном документе описывается версия 2.0 IKE; это означает, что старшая часть номера версии равна 2, а младшая часть номера версии равна 0. Вероятно, некоторые реализации захотят поддерживать обе версии 1.0 и 2.0, а в будущем и другие версии.

Старшая часть номера версии должна инкрементироваться, только если форматы пакетов или требуемые действия изменились настолько существенно, что узел со старой версией не сможет взаимодействовать с узлом с новой версией; если он просто игнорирует поля, он не понимает и выполняет действия, определенные в старой спецификации. Младшая часть номера версии указывает на новые возможности и должна (MUST) игнорироваться узлом с меньшей младшей частью номера версии, но использоваться для информационных целей узлом с большей младшей частью номера версии. Например, она может указывать на способность обрабатывать вновь определенное сообщение уведомления. Узел с большей младшей частью номера версии просто заметит, что его корреспондент не способен понимать это сообщение и, поэтому, не будет его посылать.

Если оконечная точка получает сообщение с большей старшей частью номера версии, она должна (MUST) отбросить это сообщение и должна (SHOULD) послать не аутентифицированное уведомительное сообщение, содержащее наибольший номер версии, который она поддерживает. Если оконечная точка поддерживает старшую часть номера версии n и старшую часть номера версии m, она должна (MUST) поддерживать все версии между n и m. Если она получает сообщение со старшей частью номера версии, которую она поддерживает, она должна (MUST) отвечать этим номером версии. Чтобы предотвратить возможность вовлечения двух узлов в обмен информацией с меньшей старшей частью номера версии, чем они оба поддерживают, в IKE предусмотрен флаг, который указывает на то, что узел может разговаривать с большей старшей частью номера версии.

Таким образом, старшая часть номера версии в заголовке IKE указывает номер версии сообщения, а не наибольший номер версии, который поддерживает передатчик. Если инициатор способен разговаривать на языке версий n, n+1 и n+2, а ответчик способен разговаривать на языке версий n и n+1, то они согласуют ведение разговора на языке версии n+1, при этом инициатор установит флаг, указывающий на его способность разговаривать на языке с большим номером версии. Если они по ошибке (возможно в результате действий активного атакующего злоумышленника, посылающего ошибочные сообщения) согласуют версию n, а затем оба заметят, что другая сторона может поддерживать больший номер версии, они должны (MUST) разорвать соединение и переустановить его, используя версию n+1.

Заметим, что IKEv1 не следует этим правилам, поскольку в v1 отсутствуют средства уведомления о том, что вы способны говорить на языке с большим номером версии. Таким образом, активный злоумышленник может вынудить два узла говорить на языке v1, когда они способны говорить на языке v2. Когда узел, способный говорить на языке v2, согласует понижение до v1, он должен (SHOULD) отметить этот факт в своих журналах.

Кроме того, для обеспечения совместимости снизу вверх все поля, помеченные признаком RESERVED, реализацией версии 2.0 должны (MUST) быть установлены в ноль, а их содержимое реализацией версии 2.0 должно (MUST) игнорироваться ("будьте консервативными в том, что вы посылаете, и либеральными в том, что вы принимаете"). Таким образом, будущие версии протокола могут использовать эти поля тем способом, который гарантированно будет игнорироваться реализациями, которые его не понимают. Подобным образом, не определенные в данном документе типы блоков данных резервируются для будущего использования, и реализации версии 2.0 должны (MUST) пропускать такие блоки данных и игнорировать их содержимое.

Для обеспечения совместимости снизу вверх и дополнительной гибкости IKEv2 в каждый заголовок блока данных добавляет флаг "critical" (критический). Если флаг critical установлен, а тип блока данных не распознается, то сообщение должно (MUST) быть отброшено, а ответ на IKE-запрос, содержащий такой блок данных, должен (MUST) включать блок данных Уведомление (Notify) типа UNSUPPORTED_CRITICAL_PAYLOAD, указывающий на то, что в запрос был включен неподдерживаемый критический блок данных. Если флаг critical не установлен, а данный тип блока данных не поддерживается, то такой блок данных должен (MUST) игнорироваться.

Хотя в будущем могут быть добавлены новые типы блоков данных, которые могут перемежаться с полями, определенными в данной спецификации, реализации должны (MUST) посылать определенные в данной спецификации блоки данных в порядке, показанном на рисунках раздела 2, и реализации должны (SHOULD) отклонять, как недействительное, сообщение с этими блоками данных, расположенными в любом другом порядке.

2.6. Идентифицирующие цепочки

Термин "идентифицирующие цепочки" (cookies) был введен Карном (Karn) и Симпсоном (Simpson) в протоколе Photuris [RFC2522], старом предложении по управлению ключами в IPsec, и сохранился. Фиксированный заголовок сообщений протокола управления контекстами безопасности и ключами в Internet (ISAKMP) [MSST92] включает два восьмиоктетных поля, которые названы "cookies", и этот синтаксис используется как в IKEv1, так и в IKEv2, хотя в IKEv2 они называются индексами параметров безопасности IKE (IKE SPI), а в блоке данных Уведомление (Notify) имеется новое отдельное поле, содержащее идентифицирующую цепочку. Два начальных восьмиоктетных поля заголовка используются в качестве идентификатора соединения в начале пакетов IKE. Каждая оконечная точка выбирает один из двух SPI и должна (SHOULD) выбирать их так, чтобы они были уникальными идентификаторами IKE_SA. Нулевое значение SPI является специальным и указывает на то, что значение удаленного SPI еще не известно отправителю.

В отличие от протоколов ESP и AH, в которых в заголовке сообщения появляется только SPI получателя, в IKE в каждом сообщении посылается также и SPI отправителя. Поскольку SPI, выбранный первоначальным инициатором IKE_SA, всегда посылается первым, оконечная точка с несколькими открытыми IKE_SA, желающая найти соответствующий IKE_SA с помощью SPI, который она присвоила, должна проверить бит I(nitiator) Flag в заголовке, чтобы определить, присвоила ли она первые или вторые восемь октетов.

В первом сообщении начального обмена IKE инициатор не знает значение SPI ответчика и, поэтому, установит это поле в ноль.

Одной из предполагаемых атак на IKE является исчерпание состояния и вычислительных возможностей процессора, при которой цель подвергается затоплению запросами инициализации сеанса с поддельных IP-адресов. Эту атаку можно сделать менее эффективной, если реализация ответчика минимально нагружает процессор и не фиксирует никакого состояния для SA до тех пор, пока она не узнает, что инициатор может получать пакеты на адрес, с которого по его заявлению он их посылает. Чтобы это выполнить, ответчик (когда он обнаруживает большое количество наполовину открытых IKE_SA) должен (SHOULD) отбрасывать начальные сообщения IKE, если они не содержат блока данных Уведомление (Notify) типа COOKIE. Тогда в качестве ответа он должен (SHOULD) послать незащищенное сообщение IKE и включить в него блок данных Уведомление типа COOKIE с данными идентифицирующей цепочки, которые должны быть возвращены. Инициаторы, получающие такие ответы, должны (MUST) повторить сообщение IKE_SA_INIT с блоком данных Уведомление типа COOKIE, содержащим представленные ответчиком данные идентифицирующей цепочки, который должен быть первым блоком данных в сообщении, а все другие блоки данных неизмененными. Тогда начальный обмен будет следующим:

```
Инициатор                          Ответчик
-----------                        -----------
HDR(A,0), SAi1, KEi, Ni   -->

                          <-- HDR(A,0), N(COOKIE)

HDR(A,0), N(COOKIE), SAi1, KEi, Ni   -->

                          <-- HDR(A,B), SAr1, KEr, Nr, [CERTREQ]

HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,]
    AUTH, SAi2, TSi, TSr} -->

                          <-- HDR(A,B), SK {IDr, [CERT,] AUTH,
                                        SAr2, TSi, TSr}
```

Первые два сообщения не влияют на состояние инициатора или ответчика за исключением передачи идентифицирующей цепочки. В частности, порядковые номера сообщений в первых четырех сообщениях будут нулевыми, а порядковые номера сообщений в последних двух сообщениях будут равны единице. 'A' - это SPI, присвоенный инициатором, а 'B' - SPI, присвоенный ответчиком.

Реализация IKE должна (SHOULD) реализовать генерацию идентифицирующей цепочки ответчика таким способом, чтобы, когда поступит второе сообщение IKE_SA_INIT, для распознавания своей действительной идентифицирующей цепочки не потребовалось никакого сохраненного состояния. Строгий алгоритм и синтаксис, которые они используют для генерации идентифицирующих цепочек, не влияют на интероперабельность и поэтому здесь не специфицируются. Ниже представлен пример того, как оконечная точка может использовать идентифицирующие цепочки для реализации ограниченной защиты от атак на доступность.

Хорошим способом это сделать является установка идентифицирующей цепочки ответчика равной:

```
Cookie = <VersionIDofSecret> | Hash(Ni | IPi | SPIi | <secret>)
```

где <secret> - это случайным образом генерируемый секрет, известный только ответчику и периодически меняющийся, а символ | означает конкатенацию. Идентификатор версии секрета <VersionIDofSecret> должен меняться всякий раз, когда генерируется секрет <secret>. Идентифицирующая цепочка может быть повторно вычислена, когда IKE_SA_INIT поступает во второй раз, и ее можно сравнить с идентифицирующей цепочкой в принятом сообщении. Если она совпадает, то ответчик знает, что идентифицирующая цепочка SPIr была сгенерирована после последнего изменения <secret> и что IPi должен совпадать с адресом источника, который он видел в первый раз. Включение SPIi в вычисление гарантирует, что если одновременно устанавливаются несколько IKE_SA, то все они получат различные идентифицирующие цепочки (предполагая, что инициатор выбирает уникальные SPIi). Включение в вычисление хэш-функции значения Ni гарантирует, что злоумышленник, который видит только сообщение 2, не может успешно подделать сообщение 3.

Если, несмотря на то, что существуют соединения, находящиеся в процессе инициализации, для секрета <secret> выбирается новое значение, запрос IKE_SA_INIT может быть возвращен с идентификатором версии секрета <VersionIDofSecret>, отличным от текущего. В таком случае ответчик может (MAY) отклонить сообщение путем посылки другого ответа с новой идентифицирующей цепочкой, или он может (MAY) хранить старое значение секрета (<secret>) в течение короткого времени и принимать идентифицирующие цепочки, вычисленные на основе любого из них. После изменения <secret> ответчик не должен (SHOULD NOT) принимать идентифицирующие цепочки бесконечно долго, поскольку это снимает часть защиты от атаки на доступность. Ответчик должен (SHOULD) менять значение <secret> часто, особенно находясь под воздействием атаки.

2.7. Согласование криптографических алгоритмов

Тип блока данных, получивший название "SA", определяет для устанавливаемого контекста безопасности предложение множества возможных протоколов IPsec (IKE, ESP и/или AH), а также криптографических алгоритмов, связанных с каждым протоколом.

Блок данных SA состоит из одного или нескольких предложений. Каждое предложение включает один или несколько протоколов (обычно один). Каждый протокол содержит одно или несколько преобразований, каждое из которых определяет криптографический алгоритм. Каждое преобразование включает ноль или больше атрибутов (атрибуты нужны, только если идентификатор преобразования не полностью определяет криптографический алгоритм).

Эта иерархическая структура была разработана с целью эффективного кодирования предложений для криптографических наборов, когда количество поддерживаемых наборов велико, поскольку для нескольких преобразований приемлемыми оказываются несколько значений. Ответчик должен (MUST) выбрать один набор, который может (MAY) быть любым подмножеством предложения контекста безопасности, следуя нижеприведенным правилам:

Каждое предложение содержит один или несколько протоколов. Если предложение принимается, то ответный SA должен (MUST) содержать те же самые протоколы, следующие в том же порядке, что и в предложении. Ответчик должен принять одно предложение или отказаться от всех предложений и вернуть признак ошибки. Например, если единственное предложение содержит протоколы ESP и AH, и это предложение принимается, то должны (MUST) приниматься как протокол ESP, так и протокол AH. Если протоколы ESP и AH включены в отдельные предложения, то ответчик должен принять только одно из них.

Предложение каждого протокола IPsec включает одно или несколько преобразований. Каждое преобразование включает тип преобразования. Принимаемый криптографический набор должен (MUST) включать ровно по одному преобразованию каждого типа, включенного в предложение. Например, если предложение протокола ESP включает преобразования ENCR_3DES, ENCR_AES w/keysize 128, ENCR_AES w/keysize 256, AUTH_HMAC_MD5 и AUTH_HMAC_SHA, то принятый набор должен (MUST) содержать одно из преобразований ENCR_ и одно из преобразований AUTH_. Таким образом, приемлемыми являются шесть комбинаций.

Поскольку инициатор посылает свое значение Диффи-Хеллмана в IKE_SA_INIT, он должен угадать группу Диффи-Хеллмана, которую ответчик выберет из своего списка поддерживаемых групп. Если предположение инициатора оказалось неверным, то ответчик ответит блоком данных Уведомление (Notify) типа INVALID_KE_PAYLOAD (неправильный блок данных KE), указывая в нем выбранную группу. В этом случае инициатор должен (MUST) повторить IKE_SA_INIT с уточненной группой Диффи-Хеллмана. Инициатор должен (MUST) снова предложить свое полное множество приемлемых криптографических наборов, поскольку сообщение отклонения было не аутентифицированным, а в противном случае активный злоумышленник мог бы заставить оконечные точки согласовать более слабый набор, вместо более строгого набора, который они обе предпочитают.

2.8. Переустановка контекстов безопасности

Контексты безопасности IKE, ESP и AH используют секретные ключи, которые должны (SHOULD) использоваться только в течение ограниченного периода времени и защищать ограниченный объем данных. Это ограничивает время жизни всего контекста безопасности. Когда истекает время жизни контекста безопасности, такой контекст безопасности не должен (MUST NOT) использоваться. Если имеется необходимость, могут (MAY) устанавливаться новые контексты безопасности. Повторная установка контекстов безопасности, которые заменяют контексты безопасности, время жизни которых истекает, называется "переустановкой" (rekeying).

Чтобы учесть реализации с минимальными возможностями IPsec, способность переустановки контекстов безопасности без перезапуска всего IKE_SA является факультативной возможностью. Реализация может (MAY) отклонить все запросы CREATE_CHILD_SA в рамках IKE_SA. Если время жизни SA истекло или почти истекло, и попытки переустановки, использующие описанные здесь механизмы, не увенчиваются успехом, то реализация должна (MUST) закрыть IKE_SA и любые связанные с ним CHILD_SA, а затем может (MAY) запустить новые. Реализации должны (SHOULD) поддерживать естественную переустановку контекстов безопасности, поскольку выполнение этого требования предлагает более высокую производительность и, вероятно, сокращает количество потерянных пакетов во время перехода.

Чтобы переустановить CHILD_SA в рамках существующего IKE_SA, необходимо создать новый эквивалентный SA (см. подраздел 2.17 ниже), и, когда новый контекст безопасности установится, удалить старый. Чтобы переустановить IKE_SA, необходимо установить новый эквивалентный IKE_SA (см. подраздел 2.18 ниже) с тем партнером, с которым разделяется старый IKE_SA, используя обмен CREATE_CHILD_SA в рамках существующего IKE_SA. Открытый таким образом IKE_SA наследует все CHILD_SA первоначального IKE_SA. Далее следует использовать новый IKE_SA для всех управляющих сообщений, необходимых для поддержки контекстов безопасности CHILD_SA, созданных старым IKE_SA, и удалить старый IKE_SA. Блок данных Удаление (Delete) для удаления самого себя должен (MUST) быть последним запросом, посланным через IKE_SA.

Переустановка контекстов безопасности (SA) должна (SHOULD) выполняться с упреждением, т.е. новый SA должен быть установлен до того, как истечет время жизни и станет непригодным старый SA. Между моментом времени, когда устанавливается новый SA, а старый SA станет непригодным, должно пройти достаточно времени, чтобы трафик мог переключиться на новый SA.

Различие между IKEv1 и IKEv2 заключается в том, что в IKEv1 времена жизни контекстов безопасности согласовывались. В IKEv2 каждый конец SA несет ответственность за осуществление своей собственной политики времени жизни SA и за переустановку SA, когда необходимо. Если два конца соединения имеют различные политики времени жизни, то конец с более коротким временем жизни будет завершаться раньше и запрашивать переустановку контекста безопасности. Если связка SA была неактивной в течение длительного времени, и если оконечная точка не будет инициировать SA при отсутствии трафика, то эта оконечная точка вместо переустановки SA может (MAY) принять решение закрыть SA, когда его время жизни истечет. Она должна (SHOULD) так сделать, если, начиная с того момента, когда в последний раз произошла переустановка SA, трафик отсутствовал.

Если оба конца соединения имеют одни и те же политики времени жизни, то возможно, что эти оба конца начнут инициировать переустановку контекста безопасности одновременно (что приведет к избыточным SA). Чтобы уменьшить вероятность таких ситуаций, распределение запросов на переустановку контекстов безопасности должно (SHOULD) иметь разброс (задерживаться на случайное значение времени после того, как необходимость переустановки была замечена).

Такая форма переустановки контекстов безопасности может привести к временному появлению нескольких одинаковых SA между одними и теми же парами узлов. Если имеется два SA, пригодные для приема пакетов, то узел должен (MUST) принимать входящие пакеты через любой SA. Если, несмотря на такую коллизию, создаются избыточные SA, то SA, созданный на основе наименьшего из четырех одноразовых номеров, использованных в двух обменах, должен (SHOULD) быть закрыт оконечной точкой, которая его создала.

Заметим, что IKEv2 сознательно допускает параллельные (одновременные) SA с одними и теми же селекторами трафика между общими оконечными точками. Одной из целей этого является поддержка в разных контекстах безопасности разного качества обслуживания (QoS) трафика (см. [RFC2474], [RFC2475] и подраздел 4.1 [RFC2983]). Поэтому, в отличие от IKEv1, комбинация оконечных точек и селекторов трафика не может уникально идентифицировать SA между такими оконечными точками, так что эвристика удаления SA на основе дублированных селекторов трафика, применявшаяся при переустановке контекстов безопасности в IKEv1, не должна (SHOULD NOT) использоваться.

Узел, который инициировал переживший переустановку SA, должен (SHOULD) удалить заменяемый SA после того, как установился новый SA.

Существуют временные окна (особенно при наличии потери пакетов), в которых оконечные точки могут не соглашаться с состоянием некоторого контекста безопасности. Ответчик на запрос CREATE_CHILD_SA должен (MUST) быть готов осуществлять прием сообщений по некоторому SA до посылки своего ответа на запрос создания этого контекста безопасности, так что для инициатора неоднозначность отсутствует. Инициатор может (MAY) начать посылку сообщений по некоторому SA как только он обработает ответ. Однако инициатор не может осуществлять прием сообщений по вновь созданному SA до тех пор, пока он не получит и не обработает ответ на свой запрос CREATE_CHILD_SA. Как же тогда ответчик узнает, когда можно осуществлять посылку сообщений по вновь созданному SA?

С точки зрения технической корректности и интероперабельности ответчик может (MAY) осуществлять посылку по некоторому SA как только он пошлет свой ответ на запрос CREATE_CHILD_SA. Однако в некоторых ситуациях это может привести к тому, что пакеты без необходимости будут отброшены, так что реализация может (MAY) захотеть отложить такую посылку.

Ответчик может быть уверен в том, что инициатор готов к приему сообщений по некоторому SA, если либо (1) он получил криптографически правильное сообщение по новому SA, либо (2) новый SA заменяет (переустанавливает) существующий SA, и он получает IKE-запрос закрыть замененный SA. При переустановке SA ответчик должен

(SHOULD) продолжать посылку запросов по старому SA до тех пор, пока не произойдет одно из этих событий. При установке нового SA ответчик может (MAY) задержать посылку сообщений по новому SA до тех пор, пока либо он не получит сообщение, либо не произойдет тайм-аут. Если инициатор получает сообщение по некоторому SA, для которого он не получил ответ на свой запрос CREATE_CHILD_SA, он должен (SHOULD) интерпретировать это как вероятную потерю пакета и повторно передать запрос CREATE_CHILD_SA. Инициатор при отсутствии очереди сообщений может (MAY) послать фиктивное сообщение по вновь созданному SA, чтобы убедить ответчика в том, что он готов принимать сообщения.

2.9. Согласование селекторов трафика

Когда IP-пакет принимается IPsec-подсистемой, соответствующей RFC4301, и этот пакет соответствует "защищаемому" селектору в ее базе данных политик безопасности SPD, подсистема должна (MUST) защитить этот пакет с помощью IPsec. Если еще не существует SA, то задача IKE состоит в том, чтобы его создать. Поддержка SPD системы выходит за рамки IKE (см. в [PFKEY] пример протокола), хотя некоторые реализации могут обновлять свою SPD в связи с выполнением IKE (пример сценария см. в п. 1.1.3).

Блоки данных Селекторы Трафика (TS - Traffic Selector) позволяют оконечным точкам сообщать своим партнерам некоторую информацию из своих SPD. Блоки данных TS определяют критерии отбора пакетов, которые будут пересылаться через вновь установленный SA. В некоторых ситуациях такая информация может служить для проверки соответствия, чтобы гарантировать, что SPD согласованы. В других ситуациях она управляет динамическим обновлением SPD.

В каждом сообщении обмена, который создает пару CHILD_SA, появляются два блока данных TS. Каждый блок данных TS содержит один или несколько селекторов трафика. Каждый селектор трафика включает диапазон адресов (IPv4 или IPv6), диапазон портов и идентификатор IP-протокола. При поддержке сценария, описанного в пункте 1.1.3, инициатор может запросить, чтобы ответчик присвоил IP-адрес и сообщил его.

Протокол IKEv2 позволяет ответчику выбрать подмножество трафика, предлагаемого инициатором. Это может произойти, когда конфигурации двух оконечных точек соединения обновляются, но только один конец получил новую информацию. Поскольку две оконечных точки могут конфигурироваться разными людьми, несогласованность может сохраняться достаточно продолжительный период времени даже при отсутствии ошибок. Протокол допускает также намеренно различные конфигурации, как, например, когда один из концов конфигурируется для туннелирования всех адресов и зависит от другого конца, чтобы иметь новейший список.

Первый из двух блоков TS называется селектором трафика инициатора TSi (Traffic Selector - initiator). Второй блок TS называется селектором трафика ответчика TSr (Traffic Selector - responder). TSi определяет адрес источника трафика, пересылаемого от инициатора пары CHILD_SA (или адрес места назначения трафика, пересылаемого инициатору пары CHILD_SA). TSr определяет адрес места назначения трафика, пересылаемого ответчику пары CHILD_SA (или адрес источника трафика, пересылаемого ответчиком пары CHILD_SA). Например, если первоначальный инициатор запрашивает создание пары CHILD_SA и хочет туннерировать весь трафик из подсети 192.0.1.* на стороне инициатора в подсеть 192.0.2.* на стороне ответчика, инициатор включит один селектор трафика в каждый блок данных TS. TSi будет определять диапазон адресов (192.0.1.0 - 192.0.1.255), а TSr будет определять диапазон адресов (192.0.2.0 - 192.0.2.255). Предполагая, что предложение было приемлемым для ответчика, последний пошлет назад идентичные блоки данных TS. (Примечание: диапазон IP-адресов 192.0.1.* зарезервирован для использования в примерах в RFC и подобных документах. В данном документе требуются два таких диапазона, и поэтому используется также диапазон 192.0.2.*. Их не следует путать ни с каким действительным адресом).

Ответчику разрешается ограничить возможности путем выбора некоторого подмножества трафика, например, путем выделения или ограничения диапазона одного или нескольких элементов множества селекторов трафика при условии, что множество не станет пустым.

Политика ответчика может включать несколько диапазонов меньшего размера, которые попадают в селектор трафика инициатора, и при условии, что политика ответчика заключается в том, что каждый из этих диапазонов должен посылаться через отдельный SA. Продолжая вышеприведенный пример, ответчик может иметь политику, вынуждающую туннелировать эти адреса инициатору и от него, но может потребовать, чтобы каждая пара адресов была на отдельно согласованном CHILD_SA. Если инициатор сгенерировал свой запрос в ответ на входящий пакет с адреса 192.0.1.43 на адрес 192.0.2.123, у ответчика не будет способа определить, какая пара адресов должна быть включена в этот туннель, и он будет вынужден сделать некоторое предположение или отбросить запрос с состоянием ошибки SINGLE_PAIR_REQUIRED (требуется одна пара).

Чтобы разрешить ответчику выбрать соответствующий диапазон в том случае, когда инициатор запросил SA в связи с поступлением пакета данных, инициатор должен (SHOULD) включить в качестве первого селектора трафика в каждый TSi и TSr достаточно специфические селекторы трафика, включающие адреса из пакета, инициировавшего запрос. В данном примере, инициатор включит в TSi два селектора трафика: первый селектор, содержащий диапазон адресов (192.0.1.43 - 192.0.1.43), порт источника и IP-протокол из пакета, и второй селектор, содержащий диапазон (192.0.1.0 - 192.0.1.255) со всеми портами и IP-протоколами. Подобным образом, инициатор включит два селектора трафика в TSr.

Если политика ответчика не разрешает ему принимать полный набор селекторов трафика из запроса инициатора, но позволяет ему принять первый селектор TSi и TSr, то ответчик должен (MUST) ограничить селекторы трафика до подмножества, которое включает первые альтернативы инициатора. В данном примере ответчик может ответить TSi, содержащим (192.0.1.43 - 192.0.1.43) со всеми портами и IP-протоколами.

Если инициатор создает пару CHILD_SA не в ответ на поступивший пакет, а скорее, скажем, после запуска, то для установки начального туннеля у инициатора, вероятно, отсутствуют критерии выбора конкретных адресов. В этом случае, первые значения в TSi и TSr могут (MAY) скорее указывать диапазоны, а не конкретные значения, а ответчик выбирает подмножества TSi и TSr инициатора, которые для него являются приемлемыми. Если приемлемым оказывается более одного подмножества, но их объединение является неприемлемым, ответчик должен (MUST) принять некоторое подмножество и может (MAY) включить в ответ блок данных Уведомление (Notify) типа ADDITIONAL_TS_POSSIBLE (возможны дополнительные TS), чтобы указать, что инициатор может повторить попытку. Такая ситуация может возникнуть, только если инициатор и ответчик конфигурируются по-разному. Если инициатор и ответчик договорились относительно уровней детализации туннелей, инициатор никогда не запросит более широкого туннеля, чем примет ответчик. Такое неправильное конфигурирование должно (SHOULD) быть записано в журналы ошибок.

2.10. Одноразовые номера

Каждое сообщение IKE_SA_INIT содержит одноразовый номер (nonce). Эти одноразовые номера используются в качестве входных данных криптографических функций. Запрос CREATE_CHILD_SA и ответ CREATE_CHILD_SA также содержат одноразовые номера. Эти одноразовые номера используются для того, чтобы освежить ключевой материал для метода генерации ключей, используемого для получения ключей для CHILD_SA, а также для того, чтобы гарантировать формирование строго псевдослучайной последовательности бит из ключа Диффи-Хеллмана. Одноразовые номера, используемые в IKEv2, должны (MUST) выбираться случайным образом, должны (MUST) быть размером не менее 128 бит и должны (MUST) иметь размер, равный по крайней мере половине размера ключа согласованной псевдослучайной функции prf (имя "prf" означает "pseudo-random function", один из криптографических алгоритмов, согласуемый в обмене IKE). Если для ключей и одноразовых номеров используется один и тот же источник случайных чисел, то необходимо позаботиться о том, чтобы последнее использование не подорвало первое.

2.11. Перестройка адресов и портов

Протокол IKE работает поверх UDP через порты 500 и 4500 и неявно устанавливает контексты безопасности ESP и AH для тех же самых IP-адресов, через которые он работает. Однако IP-адреса и порты во внешнем заголовке сами по себе криптографически не защищены, а IKE разработан имея в виду даже работу через устройства трансляции сетевых адресов NAT (Network Address Translation). Реализация должна (MUST) принимать поступающие запросы, даже если порт источника не равен 500 или 4500, и должна (MUST) отвечать на адрес и порт, с которых был получен запрос. Она должна установить адрес и порт, на которые был получен запрос, в качестве адреса и порта в ответе. Протокол IKE работает одинаково через IPv4 и IPv6.

2.12. Повторное использование показателей Диффи-Хеллмана

Чтобы добиться свойства "совершенной прогрессирующей секретности" (PFS - perfect forward secrecy), протокол IKE генерирует ключевой материал с помощью непродолжительного обмена Диффи-Хеллмана. Это означает, что когда соединение закрывается, и соответствующие ему ключи забываются, даже тот, кто записал все данные соединения и получил доступ ко всем долговременным ключам обеих оконечных точек, не может восстановить ключи, использовавшиеся для защиты разговора, без лобового поиска в пространстве сеансовых ключей.

Достижение совершенной прогрессирующей секретности требует, чтобы при закрытии соединения каждая оконечная точка забывала (MUST) не только ключи, использовавшиеся соединением, но и любую информацию, которая может использоваться для повторного вычисления этих ключей. В частности, она должна забыть секреты, использовавшиеся в вычислении Диффи-Хеллмана, и любое состояние, которое может сохраниться в состоянии генератора псевдослучайных чисел, которые могут использоваться для повторного вычисления секретов Диффи-Хеллмана.

Поскольку вычисление показателей Диффи-Хеллмана с точки зрения вычислительных ресурсов оказывается дорогим, оконечная точка может посчитать полезным повторное использование этих показателей для установки нескольких соединений. Имеется несколько разумных стратегий такой реализации. Оконечная точка может выбирать новый показатель только периодически, хотя это может привести к меньшей секретности, чем дает совершенная прогрессирующая секретность, если некоторое соединение продолжает существовать меньше времени жизни показателя. Или она может отслеживать, какой показатель использовался для каждого соединения, и уничтожать информацию, связанную с этим показателем, только когда некоторое соответствующее соединение было закрыто. Это позволит повторно использовать показатель без потери совершенной прогрессирующей секретности за счет поддержки большего состояния.

Решение применять ли вообще и когда применять повторное использование показателей Диффи-Хеллмана является частным решением в том смысле, что оно не влияет на интероперабельность (возможность обеспечения взаимодействия между различными реализациями). Реализация, которая повторно использует показатели, может (MAY) принять решение хранить показатель, использовавшийся другой оконечной точкой в последних обменах, и использовать его повторно, чтобы избежать второй части вычисления.

2.13. Генерация ключевого материала

В контексте IKE_SA согласуются четыре криптографических алгоритма: алгоритм шифрования, алгоритм защиты целостности, группа Диффи-Хеллмана и псевдослучайная функция (prf). Псевдослучайная функция используется для создания ключевого материала для всех криптографических алгоритмов, используемых как в контексте безопасности IKE_SA, так и в контекстах безопасности CHILD_SA.

Мы предполагаем, что каждый алгоритм шифрования и каждый алгоритм защиты целостности, использует ключ фиксированного размера, и что в качестве соответствующего ключа может служить любое случайно выбранное значение этого фиксированного размера. Для алгоритмов, которые принимают ключ переменной длины, как часть согласованного криптографического преобразования должен (MUST) быть специфицирован фиксированный размер ключа. Для тех алгоритмов, у которых не все значения являются правомерными ключами (например, DES или 3DES with key parity), криптографическим преобразованием должен (MUST) специфицироваться алгоритм, с помощью которого получаются ключи из произвольных значений. Для функций защиты целостности, базирующихся на алгоритме HMAC, фиксированным размером ключа является размер выходных данных соответствующей хэш-функции. Когда функция prf берет ключ переменной длины, данные переменной длины, и вырабатывает выходные данные фиксированной длины (например, при использовании HMAC), применяются формулы из данного документа. Когда ключ для функции prf имеет фиксированный размер, данные, предоставляемые в качестве ключа, при необходимости усекаются или дополняются нулями, за исключением случаев, когда необычная обработка не поясняется вслед за формулой.

Ключевой материал всегда будет получаться в виде выходных данных согласованного алгоритма вычисления функции prf. Поскольку объем необходимого ключевого материала может быть больше размера выходных данных алгоритма prf, мы будем использовать prf итеративно. Мы будем использовать термин prf+ для описания функции, выходом которой является псевдослучайный поток, базирующийся на входных данных prf следующим образом (где | означает конкатенацию):

```
   prf+ (K,S) = T1 | T2 | T3 | T4 | ...
```

где:

```
   T1 = prf (K, S | 0x01)
   T2 = prf (K, T1 | S | 0x02)
   T3 = prf (K, T2 | S | 0x03)
   T4 = prf (K, T3 | S | 0x04)
```

продолжая при необходимости вычислять все требуемые ключи. Ключи берутся из выходной строки, не принимая во внимание границы (например, если требуемыми ключами являются 256-битный ключ AES и 160-битный ключ HMAC, а функция pfr генерирует 160 бит, то ключ AES будет представляться из T1 и начала T2, в то время как ключ HMAC будет представляться из остатка T2 и начала T3).

Константа, конкатенируемая к концу каждой строки, поступающей на вход prf, представляет собой один октет. В данном документе функция псевдослучайного потока prf+ считается не определенной, если его длина превышает больше чем в 256 раз длину выходных данных функции prf.

2.14. Генерация ключевого материала для IKE_SA

Общие ключи вычисляются следующим образом. Величина, называемая SKEYSEED (порождающим секретным ключом) вычисляется из одноразовых номеров, обмен которыми выполняется в процессе обмена IKE_SA_INIT, и общего секрета Диффи-Хеллмана, установленного в процессе этого обмена. SKEYSEED используется для вычисления семи других секретов: SK_d, который используется для получения новых ключей для CHILD_SA, устанавливаемых в этом IKE_SA; SK_ai и SK_ar, которые используются в качестве ключей алгоритма защиты целостности для аутентификации составных сообщений последующих обменов; SK_ei и SK_er, которые используются для шифрования (и, конечно, для расшифровывания) всех последующих сообщений; и SK_pi и SK_pr, которые используются для генерации блока данных AUTH.

Ключ SKEYSEED и его производные вычисляются следующим образом:

```
       SKEYSEED = prf(Ni | Nr, g^ir)

       {SK_d | SK_ai | SK_ar | SK_ei | SK_er | SK_pi | SK_pr }
                 = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr )
```

(указывающая на то, что величины SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi и SK_pr берутся по порядку из генерируемых бит функции prf+). Величина g^ir представляет собой общий секрет непродолжительного обмена Диффи-Хеллмана. Секрет g^ir представляется строкой октетов, расположенных в порядке возрастания номеров битов, дополненных нулями, если необходимо, чтобы сделать его длину, равной длине модуля. Ni и Nr – одноразовые номера, отделенные от любых заголовков. Если согласованная prf берет ключ фиксированной длины, и длины Ni и Nr не согласуются с этой длиной, то половина бит должна браться из Ni, а другая половина из Nr, при этом берутся первые биты каждого из них.

Два направления передачи трафика используют разные ключи. Для защиты сообщений, исходящих от первоначального инициатора, используются ключи SK_ai и SK_ei. Для защиты сообщений в другом направлении используются ключи SK_ar и SK_er. Каждый алгоритм берет фиксированное количество бит ключевого материала, которое специфицируется как часть алгоритма. Для алгоритмов целостности, базирующихся на управляемой ключом хэш-функции, размер ключа всегда равен длине выходных данных соответствующей хэш-функции.

2.15. Аутентификация IKE_SA

Если не используется расширяемая аутентификация (см. подраздел 2.16), партнеры аутентифицируются путем подписи каждым из них некоторого блока данных (или путем вычисления кода MAC при использовании в качестве ключа общего секрета). Для ответчика октеты, которые должны быть подписаны, начинаются с первого октета SPI в заголовке второго сообщения и заканчиваются последним октетом последнего блока данных во втором сообщении. К ним привешивается (с целью вычисления подписи) одноразовый номер инициатора Ni (только значение, а не блок данных, содержащий его) и значение prf(SK_pr,IDr'), где IDr' представляет собой блок данных ID ответчика за исключением фиксированного заголовка. Заметим, что ни одноразовый номер Ni, ни значение prf(SK_pr,IDr') не передаются. Подобным образом, инициатор подписывает первое сообщение, начиная с первого октета SPI в заголовке и заканчивая последним октетом последнего блока данных. К ним привешивается (с целью вычисления подписи) одноразовый номер ответчика Nr и значение prf(SK_pi,IDi'). В приведенном выше вычислении IDi' и IDr' представляют собой полные блоки данных ID за исключением фиксированного заголовка. Для безопасности обмена важно, чтобы каждая сторона подписала одноразовый номер другой стороны.

Заметим, что подписываются все блоки данных, включая любые типы блоков данных, не включенные в данный документ. Если первое сообщение обмена посылается дважды (во второй раз с идентифицирующей цепочкой ответчика и/или с другой группой Диффи-Хеллмана), это вторая версия сообщения, которое подписывается.

Дополнительно (факультативно) сообщения 3 и 4 могут (MAY) включать сертификат, или цепочку сертификатов, обеспечивающих доказательство того, что ключ, используемый для вычисления цифровой подписи, принадлежит имени из блока данных ID. Подпись или код MAC будут вычисляться с помощью алгоритмов, продиктованных типом ключа, используемого подписчиком, и специфицируются полем Auth Method в блоке данных Аутентификация (Authentication). Отсутствует требование, чтобы инициатор и ответчик осуществляли подпись с помощью одного и того же криптографического алгоритма. Выбор криптографических алгоритмов зависит от типа ключа, который имеет каждый из них. В частности, инициатор может использовать общий ключ, в то время как ответчик может иметь открытый ключ подписи и сертификат. Обычно будет так (но это не требуется), что если для аутентификации используется общий секрет, то один и тот же ключ используется в обоих направлениях. Заметим, что общепринятой, но как правило небезопасной, практикой является наличие общего ключа, полностью полученного из выбранного пользователем пароля, без включения в него другого источника случайности.

Обычно это небезопасно, поскольку выбираемые пользователем пароли, вероятно, не будут иметь достаточной непредсказуемости, чтобы выдержать атаки по словарю, и такой метод аутентификации не позволяет предотвратить эти атаки. (Приложения, использующие для начальной загрузки (bootstrapping) и IKE_SA аутентификацию, основанную на паролях, должны использовать метод аутентификации из подраздела

2.16, который разработан для предотвращения автономных атак по словарю). Заранее распределенный ключ должен (SHOULD) содержать столько непредсказуемости, сколько ее содержит наиболее стойкий ключ, полученный в процессе согласования. В случае заранее распределенного ключа значение AUTH вычисляется следующим образом:

```
AUTH = prf(prf(Shared Secret,"Key Pad for IKEv2"), <msg octets>)
```

где строка "Key Pad for IKEv2" представляет собой 17 символов кода ASCII без завершающего нулевого символа. Общий секрет (shared secret) может быть переменной длины. Строка заполнения добавляется так, чтобы в случае получения общего секрета из пароля, реализация IKE не нуждалась в хранении пароля в виде открытого текста, а скорее могла хранить значение prf(Shared Secret,"Key Pad for IKEv2"), которое не может использоваться в качестве эквивалента пароля для протоколов, отличных от протокола IKEv2. Как подмечено выше, получение общего секрета из пароля не безопасно. Эта конструкция используется, поскольку предполагается, что люди все равно будут ее применять. Интерфейс управления, с помощью которого предоставляется общий секрет, должен (MUST) принимать строки ASCII длиной не менее 64 октетов, и перед их использованием в качестве общих секретов не должен (MUST NOT) добавлять нулевой завершающий символ. Он должен (MUST) также принимать шестнадцатеричную (HEX) кодировку общего секрета. Интерфейс управления может (MAY) принимать другие кодировки, если специфицирован алгоритм трансляции в двоичную строку. Если согласованная функция prf берет ключ фиксированной длины, общий секрет должен (MUST) иметь эту фиксированную длину.

2.16. Методы протокола расширяемой аутентификации (EAP)

В дополнение к аутентификации, использующей подписи с открытым ключом и общие секреты, IKE поддерживает аутентификацию, использующую методы, определенные в RFC 3748 [EAP]. Обычно эти методы являются асимметричными (разработаны для аутентификации пользователя сервером), и они не могут быть взаимными. По этой причине эти протоколы обычно используются для аутентификации инициатора ответчиком и должны (MUST) использоваться вместе с аутентификацией ответчика инициатором, базирующейся на подписи с открытым ключом. Эти методы часто связаны с механизмами, которые называются механизмами "традиционной аутентификации" (Legacy Authentication).

Хотя данный меморандум приводит в качестве ссылки [EAP] с целью показать, что в будущем могут быть добавлены новые методы без изменения данной спецификации, некоторые более простые изменения документируются прямо здесь и в подразделе 3.16. [EAP] определяет протокол аутентификации, требующий переменного количества сообщений. Расширяемая аутентификация реализуется в IKE в виде дополнительных обменов IKE_AUTH, которые должны (MUST) быть завершены, чтобы инициализировать IKE_SA.

Инициатор указывает желание использовать расширяемую аутентификацию путем опускания блока данных AUTH в сообщении 3. Путем включения блока данных IDi, а не блока данных AUTH, инициатор объявляет идентификатор, но пока его не подтвердил. Если ответчик хочет использовать метод расширяемой аутентификации, он поместит блок данных EAP в сообщение 4 и откладывает посылку SAr2, TSi и TSr до тех пор, пока не завершится аутентификация инициатора в последующем обмене IKE_AUTH. В случае расширяемой аутентификации с минимальным количеством сообщений начальное установление SA будет выглядеть следующим образом:

```
 Инициатор                          Ответчик
-----------                        -----------
 HDR, SAi1, KEi, Ni         -->

                            <--    HDR, SAr1, KEr, Nr, [CERTREQ]

 HDR, SK {IDi, [CERTREQ,] [IDr,]
          SAi2, TSi, TSr}   -->

                            <--    HDR, SK {IDr, [CERT,] AUTH,
                                          EAP }
```

```
      HDR, SK {EAP}             -->

                                <--    HDR, SK {EAP (success)}

      HDR, SK {AUTH}            -->

                                <--    HDR, SK {AUTH, SAr2, TSi, TSr }
```

Для методов EAP, которые создают общий ключ как побочный эффект аутентификации в сообщениях 5 и 6, используя синтаксис для общих секретов, определенный в подразделе 2.15, этот общий ключ должен (MUST) применяться как инициатором, так и ответчиком для генерации блоков данных AUTH. Общий ключ от EAP представляет собой поле из спецификации EAP, названное MSK. Общий ключ, сгенерированный в процессе обмена IKE, не должен MUST NOT использоваться ни для каких других целей.

Методы EAP, которые не устанавливают общий ключ, не должны (SHOULD NOT) использоваться, если эти методы EAP применяются в других протоколах, которые не используют аутентифицированного сервером туннеля, поскольку они являются предметом целого ряда атак типа "человек посередине" [EAPMITM]. Более подробно см., пожалуйста, в разделе Анализ безопасности. Если используются методы EAP, которые не генерируют общий ключ, блоки данных AUTH в сообщениях 7 и 8 должны (MUST) генерироваться с помощью SK_pi и SK_pr, соответственно.

Инициатор IKE_SA, использующий EAP, должен (SHOULD) быть способным расширить начальный протокольный обмен по крайней мере до десяти обменов IKE_AUTH в случае, когда ответчик посылает сообщения уведомления и/или делает повторные попытки подсказать аутентификацию. Когда протокольный обмен, определенный выбранным методом аутентификации EAP, успешно завершился, ответчик должен (MUST) послать блок данных EAP, содержащий сообщение Success. Подобным образом, если метод аутентификации оказался неудачным, ответчик должен (MUST) послать блок данных EAP, содержащий сообщение Failure. Ответчик может (MAY) в любой момент времени завершить обмен IKE путем посылки блока данных EAP, содержащего сообщение Failure.

Следуя такому расширенному обмену, блоки данных EAP AUTH должны (MUST) быть включены в два сообщения, следующих за сообщением, содержащим сообщение EAP Success.

2.17. Генерация ключевого материала для CHILD_SA

Одиночный контекст безопасности CHILD_SA создается обменом IKE_AUTH, а дополнительные контексты безопасности CHILD_SA могут быть факультативно созданы обменами CREATE_CHILD_SA. Ключевой материал для них генерируется следующим образом:

```
KEYMAT = prf+(SK_d, Ni | Nr)
```

Где Ni и Nr представляют собой одноразовые номера из обмена IKE_SA_INIT, если этот запрос является первым создаваемым CHILD_SA, либо обновленными Ni и Nr из обмена CREATE_CHILD_SA, если создается следующий контекст безопасности.

Для обменов CREATE_CHILD_SA, включающих факультативный обмен Диффи-Хеллмана, ключевой материал определяется так:

```
KEYMAT = prf+(SK_d, g^ir (new) | Ni | Nr )
```

где g^ir (new) представляет собой общий секрет, полученный в результате кратковременного обмена Диффи-Хеллмана данного обмена CREATE_CHILD_SA (представленный в виде строки октетов, идущих в сетевом порядке, дополненных, если необходимо, нулями в старших битах так, чтобы ее длина совпадала с длиной модуля).

Согласование одного CHILD_SA может закончиться созданием нескольких контекстов безопасности. ESP SA и AH SA возникают парами (по одному в каждом направлении), и при согласовании одного CHILD_SA могут быть созданы четыре контекста безопасности, если согласуется комбинация ESP и AH.

Ключевой материал должен (MUST) браться из расширенного KEYMAT в следующем порядке:

Все ключи для контекстов безопасности, передающих данные от инициатора ответчику, берутся до ключей для контекстов безопасности, идущих в обратном направлении.

Если согласуется несколько IPsec-протоколов, ключевой материал берется в том порядке, в котором протокольные заголовки появятся в инкапсулированном пакете.

Если один протокол имеет как ключ шифрования, так и ключ аутентификации, то ключ шифрования берется из первых октетов KEYMAT, а ключ аутентификации берется из следующих октетов.

Каждый криптографический алгоритм берет фиксированное количество бит ключевого материала, которое специфицируется как часть этого алгоритма.

2.18. Переустановка IKE_SA с помощью обмена CREATE_CHILD_SA

Обмен CREATE_CHILD_SA может использоваться для переустановки существующего IKE_SA (см. подраздел 2.8). Новые SPI инициатора и ответчика передаются в полях SPI. Блоки данных TS при переустановке IKE_SA опускаются. SKEYSEED для нового IKE_SA вычисляется с помощью SK_d из существующего IKE_SA следующим образом:

```
SKEYSEED = prf(SK_d (old), [g^ir (new)] | Ni | Nr)
```

где g^ir (new) является общим секретом из кратковременного обмена Диффи-Хеллмана данного обмена CREATE_CHILD_SA (представленным в виде строки октетов, идущих в сетевом порядке, дополненных, если необходимо, нулями в старших битах, чтобы ее длина совпадала с длиной модуля), а Ni и Nr являются двумя одноразовыми номерами, освобожденными от заголовков.

Новый IKE_SA должен (MUST) переустановить свои счетчики сообщений в 0.

Значения SK_d, SK_ai, SK_ar, SK_ei и SK_er вычисляются из SKEYSEED, как специфицировано в подразделе 2.14.

2.19. Запрос внутреннего адреса в удаленной сети

В сценарии взаимодействия оконечной точки с защитным шлюзом часто возникает ситуация, когда оконечной точке может потребоваться IP-адрес в сети, защищенной защитным шлюзом, и может потребоваться, чтобы этот адрес выделялся динамически. Запрос на такой временный адрес может быть включен в любой запрос создания CHILD_SA (включая неявный запрос в сообщении 3) путем включения блока данных CP.

Эта функция обеспечивает распределение адресов клиенту удаленного доступа IPsec (IRAC - IPsec Remote Access Client), пытающемуся организовать туннель в сеть, защищенную сервером удаленного доступа IPsec (IRAS - IPsec Remote Access Server). Поскольку обмен IKE_AUTH создает IKE_SA и CHILD_SA, IRAC должен (MUST) запросить адрес, контролируемый IRAS (и дополнительно другую информацию, касающуюся защищенной сети), в обмене IKE_AUTH. IRAS может предоставить адрес для IRAC из любого источника, например, с сервера DHCP/BOOTP или из своего собственного пула адресов.

```
Инициатор                            Ответчик
-----------------------------        ---------------------------
HDR, SK {IDi, [CERT,] [CERTREQ,]
```

```
        [IDr,] AUTH, CP(CFG_REQUEST),
        SAi2, TSi, TSr}              -->

                                     <--   HDR, SK {IDr, [CERT,] AUTH,
                                            CP(CFG_REPLY), SAr2,
                                            TSi, TSr}
```

Во всех случаях блок данных CP должен (MUST) быть вставлен до блока данных SA. В тех разновидностях протокола, в которых имеется несколько обменов IKE_AUTH, блоки данных CP должны (MUST) вставляться в сообщения, содержащие блоки данных SA.

CP(CFG_REQUEST) должен (MUST) включать по крайней мере один атрибут INTERNAL_ADDRESS (либо IPv4, либо IPv6), но может (MAY) содержать любое количество дополнительных атрибутов, которые инициатор хочет получить в ответе.

Например, сообщение от инициатора ответчику:

```
      CP(CFG_REQUEST)=
        INTERNAL_ADDRESS(0.0.0.0)
        INTERNAL_NETMASK(0.0.0.0)
        INTERNAL_DNS(0.0.0.0)
      TSi = (0, 0-65535,0.0.0.0-255.255.255.255)
      TSr = (0, 0-65535,0.0.0.0-255.255.255.255)
```

Примечание: селекторы трафика содержат (протокол, диапазон портов, диапазон адресов).

Сообщение от ответчика инициатору:

```
      CP(CFG_REPLY)=
        INTERNAL_ADDRESS(192.0.2.202)
        INTERNAL_NETMASK(255.255.255.0)
        INTERNAL_SUBNET(192.0.2.0/255.255.255.0)
      TSi = (0, 0-65535,192.0.2.202-192.0.2.202)
      TSr = (0, 0-65535,192.0.2.0-192.0.2.255)
```

Все возвращенные значения будут зависеть от реализации. Как можно видеть в вышеприведенном примере, IRAS может (MAY) также послать другие атрибуты, которые не были включены в CP(CFG_REQUEST) и может (MAY) игнорировать необязательные атрибуты, которые он не поддерживает.

Ответчик не должен (MUST NOT) посылать CFG_REPLY, если сначала он не получил от инициатора CP(CFG_REQUEST), поскольку мы не хотим, чтобы IRAS выполнял ненужный поиск конфигурационной информации, если IRAC не может обрабатывать REPLY. В случае, когда конфигурация IRAS требует, чтобы для заданного идентификатора IDi использовался CP, но IRAC не смог послать CP(CFG_REQUEST), IRAS должен (MUST) не исполнять запрос, а завершить обмен IKE ошибкой FAILED_CP_REQUIRED (требуется отсутствующий блок данных CP).

2.20. Запрос версии партнера

Партнер IKE, желающий навести справки относительно версии программного обеспечения IKE своего партнера, может (MAY) использовать представленный ниже метод. Это пример запроса конфигурации в рамках информационного (INFORMATIONAL) обмена после того, как были созданы IKE_SA и первый CHILD_SA.

Реализация IKE может (MAY) отказаться сообщать информацию о версии до аутентификации или даже после аутентификации, чтобы предотвратить "зондирование" в случае, когда известно, что какая-то реализация имеет какие-либо слабости системы безопасности. В этом случае, она должна (MUST) либо вернуть пустую строку, либо не возвращать никакого блока данных CP, если CP не поддерживается.

```
       Инициатор                                  Ответчик
```

```
      ----------------------------       -------------------------
      HDR, SK{CP(CFG_REQUEST)}      -->
                                    <--    HDR, SK{CP(CFG_REPLY)}

      CP(CFG_REQUEST)=
        APPLICATION_VERSION("")

      CP(CFG_REPLY)
        APPLICATION_VERSION("foobar v1.3beta, (c) Foo Bar Inc.")
```

2.21. Обработка ошибок

Имеется много видов ошибок, которые могут возникнуть в процессе обработки IKE. Если получен запрос, который неправильно сформатирован или неприемлем по причинам политики (например, не соответствует криптографическим алгоритмам), ответ должен (MUST) содержать блок данных Уведомление (Notify), указывающий ошибку. Если ошибка происходит за рамками контекста IKE-запроса (например, узел получает ESP-сообщения на несуществующий SPI), то узел должен (SHOULD) инициировать информационный (INFORMATIONAL) обмен с блоком данных Уведомление (Notify), описывающим проблему.

Ошибки, возникающие до того, как будет установлен криптографически защищенный IKE_SA, должны обрабатываться очень тщательно. Существует компромисс между желанием обеспечения полезной диагностики проблемы и соответствующего ответа на нее, и желанием не оказаться жертвой атаки на доступность, базирующейся на подложных сообщениях.

Если узел получает сообщение на порт 500 или 4500 UDP за рамками контекста известного ему IKE_SA (и не являющееся запросом на его создание), оно может быть результатом недавнего отказа узла. Если сообщение помечается признаком ответа, узел может (MAY) провести ревизию (проверить) подозрительное событие, но не должен (MUST NOT) отвечать. Если сообщение помечается как запрос, узел может проверить подозрительное событие и может (MAY) послать ответ. Если ответ посылается, то он должен посылаться на IP-адрес и порт, с которых оно пришло с теми же самыми скопированными значениями IKE SPI и Message ID. Ответ не должен (MUST NOT) криптографически защищаться и должен (MUST) содержать блок данных Уведомление (Notify), указывающий INVALID_IKE_SPI (неправильный IKE_SPI).

Узел, получающий такой незащищенный блок данных Уведомление (Notify) не должен (MUST NOT) отвечать и не должен (MUST NOT) менять состояния любого существующего SA. Сообщение может быть подделкой или может быть ответом, который истинный корреспондент был вынужден послать в результате обмана. Узел должен (SHOULD) обрабатывать такое сообщение (а также сетевое сообщение типа ICMP destination unreachable) как намек на то, что могут существовать проблемы с SA на этот IP-адрес, и должен (SHOULD) инициировать проверку работоспособности для любого такого IKE_SA. Реализация должна (SHOULD) ограничивать частоту таких проверок, чтобы избежать вовлечения в атаку на доступность.

Узел, принимающий подозрительное сообщение с IP-адреса, с которым он имеет IKE_SA, может (MAY) послать блок данных IKE Notify в информационном (INFORMATIONAL) обмене IKE через этот SA. Получатель в результате не должен (MUST NOT) изменять состояния какого-либо SA, но должен (SHOULD) проверить событие для поддержки диагностирования неправильного функционирования. Узел должен (MUST) ограничивать скорость, с которой он будет посылать сообщения в ответ на незащищенные сообщения.

2.22. IPComp

Использование IP-компрессии [IPCOMP] может быть согласовано как часть установки CHILD_SA. Хотя IP-компрессия в каждый заголовок вставляет дополнительный заголовок, а также индекс параметров компрессии CPI (compression parameter index), виртуальный "контекст компрессии" (compression association) не существует вне контекста безопасности ESP SA или AH SA, который его содержит.

Контексты компрессии исчезают, когда заканчивается время жизни соответствующего ESP SA или AH SA, и не явно упоминаются в любом блоке данных Удаление (DELETE).

Согласование IP-компрессии отделяется от согласования криптографических параметров, связанных с CHILD_SA. Узел, запрашивающий CHILD_SA, может (MAY) объявить о своей поддержке одного или нескольких алгоритмов компрессии с помощью одного или нескольких блоков данных Notify типа IPCOMP_SUPPORTED. Ответ может (MAY) указывать признание одного алгоритма компрессии при помощи блока данных Notify типа IPCOMP_SUPPORTED. Эти блоки данных не должны (MUST NOT) появляться в сообщениях, которые не содержат блоков данных SA.

Хотя была дискуссия относительно того, чтобы позволить принимать несколько алгоритмов компрессии и чтобы иметь возможность использования разных алгоритмов для разных направлений CHILD_SA, реализации данной спецификации не должны (MUST NOT) принимать алгоритм IPComp, который не предлагался, не должны (MUST NOT) принимать более одного алгоритма и не должны (MUST NOT) осуществлять компрессию, используя алгоритм, отличающийся от предложенного и принятого при установке CHILD_SA.

Побочный эффект отделения согласования IPComp от согласования криптографических параметров заключается в том, что невозможно предлагать несколько криптографических наборов и предлагать IP-компрессию с одними из них, но не с другими.

2.23. Пересечение NAT

Шлюзы NAT (Network Address Translation) считаются сомнительными устройствами. В этом подразделе коротко описано, чем они являются и как они, вероятно, воздействуют на IKE-трафик. Многие люди считают, что устройства NAT вредны, и что мы не должны разрабатывать наши протоколы так, чтобы они работали лучше. Протокол IKE2 специфицирует некоторые интуитивные правила обработки таким образом, чтобы устройства NAT вероятнее всего могли работать.

Устройства NAT существуют главным образом благодаря нехватке IPv4-адресов, хотя имеются и другие логические обоснования. IP-узлы, которые находятся "за" NAT, имеют IP-адреса, которые не являются глобально уникальными, а скорее присваиваются из некоторого пространства, которое является уникальным в рамках сети, находящейся за NAT, но которые вероятно должны повторно использоваться узлами, находящимися за другими NAT. В общем случае, узлы, находящиеся за NAT, могут обмениваться информацией с другими узлами, находящимися за тем же самым NAT, и с узлами с глобально уникальными адресами, но не с узлами, находящимися за другими NAT. Имеются и исключения из этого правила. Когда такие узлы осуществляют соединение с узлами в реальной сети Internet, шлюз NAT "транслирует" IP-адрес источника в адрес, который будет маршрутизироваться назад на этот шлюз. В сообщениях, поступающих на шлюз из Internet, адреса назначения "транслируются" во внутренний адрес, который будет маршрутизировать пакет на правильный оконечный узел.

Устройства NAT разрабатываются так, чтобы они были "прозрачными" для оконечных узлов. Для обмена информацией через NAT не требуется какой-либо модификации программного обеспечения узла, находящегося за NAT, или узла в Internet. Достижение этой прозрачности для одних протоколов оказывается более трудным, чем для других. Протоколы, включающие IP-адреса оконечных точек в блоки данных пакета, не смогут работать, если шлюз NAT не понимает протокол и не модифицирует внутренние указатели, а также указатели в заголовках. Такое знание, по сути, является ненадежным, нарушает принципы организации сетевого уровня и часто приводит к тонким проблемам.

Открытие соединения IPsec через NAT создает дополнительные проблемы. Если соединение работает в транспортном режиме, то изменение IP-адресов в пакетах приведет к неправильному вычислению контрольных сумм, а NAT не может корректировать контрольные суммы, поскольку они криптографически защищены. Даже в туннельном режиме имеются проблемы маршрутизации, поскольку прозрачная трансляция адресов пакетов AH и ESP требует специальной логики в NAT, а эта логика является эвристической и ненадежной по своей природе. По этой причине IKEv2 может согласовать UDP-инкапсуляцию пакетов IKE и ESP. Это кодирование немного менее эффективно, но проще для обработки NAT. Кроме того, межсетевые экраны могут быть сконфигурированы для передачи IPsec-трафика через UDP, но не через ESP/AH или наоборот.

Общепринятой практикой для NAT является трансляция номеров портов TCP и UDP, а также адресов и использование номеров портов входящих пакетов для принятия решения о том, какой внутренний узел должен получить данный пакет. По этой причине, даже хотя пакеты IKE должны (MUST) посылаться с порта 500 UDP и на порт 500 UDP, они должны (MUST) быть приняты, с какого бы порта они не были бы посланы, а ответы должны посылаться на тот порт, с которого они поступили. Это требование возникает потому, что порты могут быть модифицированы при прохождении пакетов через NAT. Подобным образом, в общем случае IP-адреса оконечных точек IKE не включаются в блоки данных IKE, поскольку блоки данных криптографически защищаются и не могут прозрачно модифицироваться NAT.

Порт 4500 зарезервирован для ESP и IKE, инкапсулированных в UDP. При работе через NAT обычно лучше передавать пакеты IKE через порт 4500, поскольку некоторые старые NAT обрабатывают трафик IKE на порте 500 более искусно, пытаясь обеспечить прозрачную установку IPsec-соединения между оконечными точками, которые сами не обрабатывают пересечения NAT. Такие устройства NAT могут повлиять на представленный в данном документе механизм прямого пересечения NAT, так что оконечная точка IPsec, которая между собой и своим корреспондентом обнаруживает NAT, должна (MUST) посылать весь последующий трафик на порт 4500 и с порта 4500, для которого в устройствах NAT не должна предусматриваться специальная обработка (которая может осуществляться на порту 500).

Конкретные требования для поддержки пересечения NAT [RFC3715] указаны ниже. Поддержка пересечения NAT является факультативной возможностью. Только в данном подразделе требования, указанные как MUST, применяются только к реализациям, поддерживающим пересечение NAT.

IKE должен (MUST) прослушивать порт 4500, а также порт 500. IKE должен (MUST) отвечать на IP-адрес и порт, с которых прибыли пакеты.

Как инициатор, так и ответчик IKE должны (MUST) включать в свои пакеты IKE_SA_INIT блоки данных Notify типа NAT_DETECTION_SOURCE_IP и NAT_DETECTION_DESTINATION_IP. Эти блоки данных могут использоваться для определения того, имеется ли между хостами NAT, и какой конец находится за NAT. Местоположение этих блоков данных в пакетах IKE_SA_INIT находится сразу после блоков данных Ni и Nr (до необязательного блока данных CERTREQ).

Если ни один из полученных блоков данных NAT_DETECTION_SOURCE_IP (определение нахождения IP-адреса источника за NAT) не соответствует хэш-значению IP-адреса и порта источника, полученных из IP-заголовка пакета, содержащего этот блок данных, то это означает, что другой конец соединения находится за NAT (т.е. кто-то на пути изменил адрес источника оригинального пакета, чтобы сопоставить ему адрес устройства NAT). В этом случае, как описано позже, этот конец соединения должен допускать динамическое обновление IP-адреса другого конца соединения.

Если полученный блок данных NAT_DETECTION_DESTINATION_IP (определение нахождения IP-адреса назначения за NAT) не соответствует хэш-значению IP-адреса и порта назначения, полученных из IP-заголовка пакета, содержащего этот блок данных, то это означает, что этот конец соединения находится за NAT. В этом случае этот конец соединения должен (SHOULD) начать посылку пакетов подтверждения работоспособности (keepalive), как объяснено в [Hutt05].

Инициатор IKE должен (MUST) проверить эти блоки данных на предмет наличия и на предмет не соответствия адресам во внешнем пакете, и должен (MUST) туннелировать все будущие пакеты IKE и ESP, ассоциированные с данным IKE_SA через порт 4500 UDP.

Для туннелирования пакетов IKE через порт 4500 UDP, IKE-заголовок предваряется четырьмя нулевыми октетами, а результат непосредственно следует за заголовком UDP. Для туннелирования пакетов ESP через порт 4500 UDP, ESP-заголовок непосредственно следует за заголовком UDP. Поскольку первые четыре байта заголовка ESP содержат SPI, а правильный SPI не может быть нулевым, всегда существует возможность различить сообщения ESP и IKE.

Оригинальные IP-адреса источника и места назначения, требуемые для изменения контрольной суммы пакетов TCP и UDP транспортного режима (см. [Hutt05]), получаются из селекторов трафика, ассоциированных с обменом. В случае пересечения NAT селекторы трафика должны (MUST) содержать ровно один IP-адрес, который затем используется как оригинальный IP-адрес.

Имеются случаи, когда устройство NAT принимает решение ликвидировать еще "живые" отображения (например, в случае слишком длинного интервала между пакетами подтверждения работоспособности (keepalive), или в случае, когда устройство NAT перезагружается). Чтобы восстановиться в этих случаях, хосты, которые не находятся за NAT, должны (SHOULD) посылать все пакеты (включая пакеты повторной передачи) на IP-адрес и порт из последнего правильно аутентифицированного пакета от другого конца соединения (т.е. динамически обновлять адрес). Хост, находящийся за NAT, не должен (SHOULD NOT) этого делать, поскольку это открывает возможность для атаки на доступность. Любой аутентифицированный пакет IKE или любой аутентифицированный пакет ESP, инкапсулированный в UDP, может использоваться для обнаружения того, что IP-адрес или порт изменились.

Заметим, что подобные, но возможно не идентичные действия, вероятно, потребуются для того, чтобы обеспечить работу IKE с Mobile IP, но такая обработка в данном документе не рассматривается.

2.24. Явное уведомление о перегрузке (ECN)

Когда поведение IPsec-туннелей соответствует первоначально специфицированному в [RFC2401], использование явного уведомления о перегрузке сети ECN (Explicit Congestion Notification) не соответствует внешним IP-заголовкам, поскольку обработка, связанная с разборкой (decapsulation) туннелей отбрасывает признаки перегрузки ECN в ущерб сети. Поддержка ECN для IPsec-туннелей, базирующихся на IKEv1, требует нескольких режимов работы и согласования (см. [RFC3168]). Протокол IKEv2 упрощает эту ситуацию, требуя, чтобы ECN был пригоден для употребления во внешних IP-заголовках всех IPsec SA туннельного режима, созданных IKEv2. В частности, средства инкапсуляции и разборки туннеля для всех контекстов безопасности туннельного режима, созданных IKEv2, должны (MUST) поддерживать для туннелей опцию полной функциональности ECN, специфицированную в [RFC3168], и должны (MUST) реализовывать обработку, связанную с туннельной инкапсуляцией и разборкой, специфицированную в [RFC4301], чтобы предотвратить отбрасывание признаков перегрузки ECN.

3. Форматы заголовков и блоков данных

3.1. Заголовок IKE

Сообщения IKE используют порты 500 и/или 4500 UDP, по одному сообщению IKE на каждую дейтаграмму. Информация, которая размещается с начала пакета по UDP-заголовок, почти полностью игнорируется за исключением того, что IP-адреса и порты UDP из заголовков меняются местами и используются в возвращаемых пакетах. При посылке на порт 500 UDP сообщения IKE начинаются непосредственно вслед за заголовком UDP. При посылке на порт 4500 перед сообщениями IKE вставляются четыре нулевых октета. Эти четыре нулевых октета не являются частью сообщения IKE и не включаются ни в какие поля длины или контрольных сумм, определенные IKE. Каждое сообщение IKE начинается с заголовка IKE, обозначаемого в данном меморандуме HDR. Вслед за заголовком идут один или несколько блоков данных IKE, каждый из которых идентифицируется полем "Next Payload" (следующий блок данных) в предыдущем блоке данных. Блоки данных обрабатываются в том порядке, в котором они появляются в сообщении IKE, путем вызова подпрограммы обработки, соответствующей полю "Next Payload" в самом блоке данных IKE до тех пор, пока нулевое поле "Next Payload" не укажет на то, что следующие блоки данных отсутствуют. Если обнаруживается блок данных типа "Шифр" (Encrypted), этот блок данных расшифровывается, и его содержимое разбирается на наличие дополнительных блоков данных. Блок данных Шифр должен (MUST) быть последним блоком данных в пакете и блок данных Шифр не должен (MUST NOT) содержать другой блок данных Шифр.

Поля индексов параметров безопасности (Recipient SPI) в заголовке определяют экземпляр контекста безопасности IKE. Поэтому один экземпляр IKE может мультиплексировать отдельные сеансы с несколькими партнерами.

Все многооктетные поля, представляющие целые числа, следуют в обратном порядке байтов (иначе называется порядком "старший байт первым" или сетевым порядком байтов).

Формат заголовка IKE показан на рис. 4.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                       IKE_SA Initiator's SPI                  !
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                       IKE_SA Responder's SPI                  !
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Next Payload ! MjVer ! MnVer ! Exchange Type !     Flags     !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                          Message ID                           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                            Length                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 4. Формат заголовка IKE

- Initiator's SPI – индекс параметров безопасности инициатора (8 октетов) – Значение, выбранное инициатором для распознавания уникального контекста безопасности IKE (IKE security association). Это значение не должно (MUST NOT) быть нулевым.

- Responder's SPI - индекс параметров безопасности ответчика (8 октетов) – Значение, выбранное ответчиком для распознавания уникального контекста безопасности IKE (IKE security association). Это значение должно (MUST) быть нулевым в первом сообщении обмена IKE Initial Exchange (включая повторения этого сообщения, содержащие идентифицирующую цепочку) и не должно (MUST NOT) быть нулевым в любом другом сообщении.

- Next Payload – следующий блок данных (1 октет) – Указывает тип блока данных, который следует непосредственно за заголовком. Формат и значение каждого блока данных определяются ниже.

- Major Version – старшая часть номера версии (4 бита) – Указывает старшую часть номера версии используемого протокола IKE. Реализации, базирующиеся на данной версии IKE, должны (MUST) устанавливать поле Major Version равным 2. Реализации, базирующиеся на предыдущих версиях IKE и ISAKMP должны (MUST) устанавливать поле Major Version равным 1. Реализации, базирующиеся на данной версии IKE должны (MUST) отбрасывать или игнорировать сообщения, содержащие номер версии, больший 2.

- Minor Version – младшая часть номера версии (4 бита) – Указывает младшую часть номера версии используемого протокола IKE. Реализации, базирующиеся на данной версии IKE, должны (MUST) устанавливать поле Minor Version равным 0. Они должны (MUST) игнорировать младшую часть номера версии принимаемых сообщений.

- Exchange Type – тип обмена (1 октет) – Указывает тип используемого обмена. Он ограничивает блоки данных, посылаемые в каждом сообщении, а также порядок сообщений в обмене.

Exchange Type (тип обмена)	Значение
RESERVED (зарезервировано)	0-33
IKE_SA_INIT (инициация IKE_SA)	34
IKE_AUTH (аутентификация IKE)	35
CREATE_CHILD_SA (создание дочернего SA)	36
INFORMATIONAL (информационный)	37
RESERVED TO IANA (зарезервировано для IANA)	38-239
Reserved for private use (зарезервировано для частного использования)	240-255

- Flags – флаги (1 октет) – указывает конкретные опции, которые устанавливаются для сообщения. Наличие опций указывается установкой соответствующего бита в поле флагов. Порядок битов определяется в виде "наименее значимый бит - первым", так что бит 0 будет наименее значимым битом октета флагов. В нижеприведенном описании "установленный" бит означает, что его значение равно 1, а "сброшенный" бит – что его значение равно 0.

  - X (зарезервированы) (биты 0-2) – Эти биты должны (MUST) быть сброшены при посылке и должны (MUST) игнорироваться при приеме.

  - I(nitiator)- инициатор (бит 3 поля Flags) – Этот бит должен (MUST) устанавливаться в сообщениях, посылаемых настоящим инициатором IKE_SA, и должен (MUST) сбрасываться в сообщениях, посылаемых настоящим ответчиком. Он используется приемником для определения того, какие восемь октетов SPI были сгенерированы приемником.

  - V(ersion) – версия (бит 4 поля Flags) – Этот бит указывает, что передатчик может говорить с большей старшей частью номера версии протокола, чем та, которая указана в поле старшей части номера версии. Реализации IKEv2 должны обнулять этот бит при посылке и должны (MUST) игнорировать его в поступающих сообщениях.

  - R(esponse) – ответ (бит 5 поля Flags) – Этот бит указывает, что это сообщение является ответом на сообщение, содержащее тот же самый идентификатор сообщения. Этот бит должен (MUST) сбрасываться во всех сообщениях запроса и должен (MUST) устанавливаться во всех ответах. Оконечная точка IKE не должна (MUST NOT) генерировать ответ на сообщение, которое помечено признаком ответа.

  - X(зарезервировано) (биты 6-7 поля Flags) – Эти биты должны (MUST) быть сброшены при посылке и должны (MUST) игнорироваться при приеме.

- Message ID (4 октета) – Идентификатор сообщения, используемый для контроля повторных передач потерянных пакетов и для сопоставления запросов с ответами. Для безопасности протокола он является существенным, поскольку используется для предотвращения атак повторного воспроизведения сообщений. См. подразделы 2.1 и 2.2.

- Length – длина (4 октета) – Длина в октетах всего сообщения (заголовок плюс блоки данных).

3.2. Общий заголовок блоков данных

Каждый блок данных IKE, определенный в подразделах с 3.3 по 3.16 начинается с общего заголовка, показанного на рис. 5. Рисунки каждого описанного ниже блока данных будут включать этот общий заголовок блоков данных, но для краткости описание каждого поля будет опускаться.

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 5. Общий заголовок блоков данных.

Поля общего заголовка блоков данных определяются следующим образом:

- o Next Payload – следующий блок данных (1 один октет) – Идентификатор типа блока данных для следующего в сообщении блока данных. Если текущий блок данных является последним в сообщении, то это поле будет нулевым. Это поле обеспечивает возможность "сцепления", при которой дополнительные блоки данных могут быть добавлены в сообщение путем их присоединения к концу сообщения и установки поля "Next Payload" предыдущего блока данных для указания нового типа блока данных. Блок данных Шифр, который всегда должен быть последним блоком сообщения, является исключением. Он содержит структуры данных в формате дополнительных блоков данных. В заголовке блока данных Шифр поле Next Payload устанавливается равным типу первого включенного в него блока данных (вместо 0).

  Значения типов блоков данных

Тип следующего блока данных	Обозначение	Значение
No Next Payload (Следующий блок данных отсутствует)		0
RESERVED (зарезервированы)		1-32
Security Association (Контекст Безопасности)	SA	33
Key Exchange (Обмен Ключами)	KE	34
Identification - Initiator (Идентификация – инициатор)	IDi	35
Identification - Responder (Идентификация – ответчик)	IDr	36
Certificate (Сертификат)	CERT	37
Certificate Request (Запрос Сертификата)	CERTREQ	38
Authentication (Аутентификация)	AUTH	39
Nonce (Одноразовый Номер)	Ni, Nr	40
Notify (Уведомление)	N	41
Delete (Удаление)	D	42
Vendor ID (Идентификатор Поставщика)	V	43
Traffic Selector - Initiator (Селектор Трафика – инициатор)	TSi	44
Traffic Selector - Responder (Селектор Трафика – ответчик)	TSr	45
Encrypted (Шифр)	E	46
Configuration (Конфигурирование)	CP	47
Extensible Authentication (Расширяемая Аутентификация)	EAP	48
RESERVED TO IANA (Зарезервированы для IANA)		49-127
PRIVATE USE (Для частного использования)		128-255

  Значения типов блоков данных 1-32 не должны использоваться, так что пересечение с назначением кода для IKEv1 отсутствует. Значения типов блоков данных 49-127 зарезервированы для IANA для будущего назначения в IKEv2 (см. раздел 6). Значения типов блоков данных 128-255 предназначены для частного использования между взаимно согласными сторонами.

- o Critical – критический (1 бит) – должен (MUST) устанавливаться в ноль, если отправитель хочет, чтобы получатель опустил этот блок данных, если он не понимает код типа блока данных в поле Next Payload предыдущего блока данных. Должен (MUST) устанавливаться в единицу, если отправитель хочет, чтобы получатель отбросил все это сообщение, если тот не понимает типа блока данных. Должен (MUST) игнорироваться получателем, если получатель понимает код типа блока данных. Должен (MUST) устанавливаться в ноль для типов блоков данных, определенных в данном документе. Заметим, что бит critical применяется к текущему блоку данных, а не к "следующему" блоку данных, код типа которого появляется в первом октете. Аргументация в пользу того, чтобы не устанавливать бит critical для блоков данных, определенных в данном документе, заключается в том, что все реализации должны (MUST) понимать все типы блоков данных, определенные в данном документе, и, поэтому, должны игнорировать значение бита Critical. Предполагается, что опущенные блоки данных имеют правильные значения полей Next Payload и Payload Length.

- o RESERVED – зарезервировано (7 бит) – должны (MUST) посылаться нулевыми; должны (MUST) игнорироваться при получении.

- o Payload Length – длина блока данных (2 октета) – Длина в октетах текущего блока данных, включая общий заголовок блоков данных.

3.3. Блок данных Контекст Безопасности

Блок данных Контекст Безопасности (Security Association Payload), обозначаемый в данном меморандуме SA, используется для согласования атрибутов контекста безопасности. Сборка блоков данных Контекст Безопасности требует большого душевного спокойствия. Блок данных SA может (MAY) включать несколько предложений. Если предложений больше одного, они должны (MUST) быть упорядочены в порядке убывания предпочтительности. Каждое предложение может включать несколько протоколов IPsec (где под протоколом понимается IKE, ESP или AH), каждый протокол может (MAY) включать несколько преобразований, а каждое преобразование может (MAY) включать несколько атрибутов. При разборе SA реализация должна (MUST) проверить, что общая длина блока данных согласована с длиной и количеством внутренних блоков данных. Каждое Предложение, каждое Преобразование и каждый Атрибут имеют свою собственную кодировку переменной длины. Они являются вложенными, так что длина блока данных SA включает объединенное содержимое данных SA, предложений, преобразований и атрибутов. Длина Предложения включает длину всех Преобразований и Атрибутов, которые оно содержит. Длина Преобразования включает длину всех содержащихся в нем Атрибутов.

Синтаксис Контекстов Безопасности, Предложений, Преобразований и Атрибутов базируется на ISAKMP, однако семантика кое в чем отличается. Причина сложности и иерархии заключается в том, чтобы предоставить возможность кодирования в одном SA нескольких возможных комбинаций алгоритмов. Иногда имеется возможность выбора одного из нескольких алгоритмов, тогда как в других случаях - комбинации алгоритмов. Например, инициатор может захотеть предложить использовать (AH w/MD5 and ESP w/3DES) OR (ESP w/MD5 and 3DES).

Одной из причин того, что семантика блока данных SA поменялась по сравнению с семантикой ISAKMP и IKEv1, заключалась в том, чтобы в общих случаях сделать кодирование более компактным.

Структура Предложение (Proposal) включает в себя номер предложения (Proposal #) и идентификатор протокола IPsec (IPsec protocol ID). Каждая структура должна (MUST) иметь тот же самый номер предложения, что и предыдущая, или на единицу (1) больший номер. Первая структура Предложение должна (MUST) иметь номер предложения один (1). Если две последовательные структуры имеют один и тот же номер предложения, это означает, что предложение состоит из первой структуры AND (логическое И) второй структуры. Таким образом, предложение AH AND ESP будет иметь две предлагаемые структуры, одну для AH и одну для ESP, и обе будут иметь номер предложения 1. Предложение AH OR ESP будет иметь две предлагаемые структуры, одну для AH с номером предложения 1 и одну для ESP с номером предложения 2.

За каждой структурой Предложение/Протокол следует одна или несколько структур Преобразование. Количество различных преобразований в общем случае определяется протоколом. Как правило, AH имеет одно преобразование: алгоритм проверки целостности. Как правило, ESP имеет два преобразования: алгоритм шифрования и алгоритм проверки целостности. Как правило, IKE имеет четыре преобразования: группу Диффи-Хеллмана, алгоритм проверки целостности, алгоритм prf и алгоритм шифрования. Если предлагается алгоритм, который объединяет шифрование и защиту целостности, то он должен (MUST) предлагаться как алгоритм шифрования, а алгоритм защиты целостности предлагаться не должен (MUST NOT). Для каждого протокола множеству допустимых преобразований присваиваются номера идентификаторов преобразований, которые появляются в заголовке каждого преобразования.

Если имеется несколько преобразований с одним и тем же типом преобразования (Transform Type), то предложение является логическим ИЛИ (OR) этих преобразований. Если имеется несколько преобразований с различными типами преобразований, то предложение является логическим И (AND) различных групп. Например, чтобы предложить ESP с (3DES or IDEA) and (HMAC_MD5 or HMAC_SHA), предложение ESP будет содержать двух кандидатов с типом преобразования 1 (один для 3DES и один для IDEA) и двух кандидатов с типом преобразования 2 (один для HMAC_MD5 и один для HMAC_SHA). Такая структура в действительности предлагает четыре комбинации алгоритмов. Если инициатор желает предложить только подмножество этих преобразований – скажем (3DES and HMAC_MD5) or (IDEA and HMAC_SHA), то отсутствует способ закодировать такую структуру в одном предложении в виде нескольких преобразований. Вместо этого, инициатор должен будет создать две разных структуры Предложение, каждую с двумя преобразованиями.

Заданное преобразование может (MAY) иметь один или несколько атрибутов. Атрибуты необходимы, когда преобразование может использоваться более чем одним способом, например, когда алгоритм шифрования имеет переменную длину ключа. Преобразование будет определять алгоритм, а атрибут – размер ключа. Большинство преобразований не имеют атрибутов. Преобразование не должно (MUST NOT) иметь несколько атрибутов одного и того же типа. Чтобы для атрибутов предложить дополнительные значения (например, несколько размеров ключа для алгоритма шифрования AES), реализация должна (MUST) включить несколько структур Преобразование с одним и тем же типом преобразования, каждая из которых имеет один атрибут.

Заметим, что семантика преобразований и атрибутов совершенно отличается от семантики IKEv1. В IKEv1 в одном преобразовании передавалось несколько алгоритмов для протокола, при этом один алгоритм, передавался в преобразовании, а другие - в атрибутах.

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                          <Proposals>                          ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис.6. Блок данных Контекст Безопасности

o Proposals – предложения (переменной длины) – одна или несколько структур Предложение.

  Тип блока данных Контекст Безопасности имеет значение тридцать три (33).

3.3.1. Структура Предложение

```
                     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 (last) or 2 !   RESERVED    !         Proposal Length       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Proposal #    !  Protocol ID  !    SPI Size   !# of Transforms!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
~                         SPI (variable)                        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                        <Transforms>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис.7. Структура Предложение

- 0 (last) or 2 (more) – 0 (последняя) или 2 (не последняя) (1 октет) – Определяет, является ли эта структура Предложение последней в SA. Этот синтаксис унаследован из ISAKMP, но он необязателен, поскольку последнее предложение может быть определено исходя из длины SA. Значение (2) соответствует типу блока данных Предложение в IKEv1, и первые четыре октета структуры Предложение разработаны так, чтобы они выглядели подобно заголовку блока данных.

- RESERVED – зарезервировано (1 октет) – должен (MUST) посылаться как нулевой; должен (MUST) игнорироваться при приеме.

- Proposal Length – длина предложения (2 октета) – Длина данного предложения, включая все преобразования и атрибуты, которые за ним следуют.

- Proposal # - Номер предложения (1 октет) – Когда делается предложение, первое предложение в блоке данных SA должно (MUST) иметь номер 1, а последующие предложения должны (MUST) либо иметь тот же самый номер предложения, что и предыдущее предложение (указывая логическое И (AND) двух предложений), или на единицу больший номер, чем у предыдущего предложения (указывая на логическое ИЛИ (OR) двух предложений). Когда предложение принимается, все номера предложений в блоке данных SA должны (MUST) быть теми же самыми и должны (MUST) соответствовать количеству посланных предложений, которые были приняты.

- Protocol ID – идентификатор протокола (1 октет) – Определяет идентификатор протокола IPsec для текущего согласования. Определены следующие значения:

Протокол	Идентификатор протокола
RESERVED (зарезервировано)	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA (зарезервировано для IANA)	4-200
PRIVATE USE (для частного использования)	201-255

- SPI Size – длина SPI (1 октет) – Для начального согласования IKE_SA это поле должно (MUST) быть нулевым; SPI извлекается из внешнего заголовка. В процессе последующих согласований оно равно в октетах длине SPI соответствующего протокола (8 для IKE, 4 для ESP и AH).

- # of Transforms – количество преобразований (1 октет) – Определяет количество преобразований в данном предложении.

- SPI – индекс параметров безопасности (переменной длины) – SPI посылающей сущности. Даже если поле SPI Size не кратно 4 октетам, к блоку данных заполнение не применяется. Если значение поля SPI Size равно нулю, то это поле в блоке данных Контекст Безопасности отсутствует.

- Transforms – преобразования (поле переменной длины) – одна или несколько подструктур Преобразование.

3.3.2. Подструктура Преобразование

```
                     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 (last) or 3 !   RESERVED    !        Transform Length       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!Transform Type !   RESERVED    !          Transform ID         !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                      Transform Attributes                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 8. Подструктура Преобразование.

- 0 (last) or 3 (more) – 0 (последняя) или 3 (не последняя) (1 октет) – Определяет, является ли данная подструктура Преобразование последней в структуре Предложение. Этот синтаксис унаследован из ISAKMP, но является необязательным, поскольку последнее преобразование может быть определено из длины SA. Значение (3) соответствует типу блока данных Преобразование в IKEv1, а первые четыре октета структуры Преобразование разработаны так, чтобы выглядеть подобно заголовку блока данных.

- RESERVED – (зарезервировано) – должно (MUST) посылаться нулем; должно (MUST) игнорироваться при приеме.

- Transform Length – длина преобразования – Длина (в октетах) подструктуры Преобразование, включая заголовок и атрибуты.

- Transform Type – тип преобразования - (1 октет) – Тип преобразования, определенный в данном преобразовании. Различные протоколы поддерживают разные типы преобразований. Для некоторых протоколов некоторые преобразования могут быть необязательными. Если преобразование является необязательным и инициатор хочет предложить, чтобы преобразование было опущено, в предложение не включается никаких преобразований данного типа. Если инициатор хочет воспользоваться необязательным для ответчика преобразованием, он включает подструктуру преобразования с идентификатором преобразования, равным 0, как одну из опций.

- Transform ID – идентификатор преобразования - (2 октета) – Конкретный экземпляр предлагаемого типа преобразования.

  Значения типов преобразований

	Тип преобразования	Используется в
RESERVED (зарезервировано)	0	
Encryption Algorithm (ENCR) (алгоритм шифрования)	1	(IKE и ESP)
Pseudo-random Function (PRF) (псевдослучайная функция)	2	(IKE)
Integrity Algorithm (INTEG) (алгоритм целостности)	3	(IKE, AH, опция в ESP)
Diffie-Hellman Group (D-H) (группа Диффи-Хеллмана)	4	(IKE, опция в AH & ESP)
Extended Sequence Numbers (ESN) (расширенные порядковые номера)	5	(опция в AH и ESP)
RESERVED TO IANA (зарезервировано для IANA)	6-240	
PRIVATE USE (для частного использования)	241-255	

Для типа преобразования 1 (алгоритм шифрования) определены следующие идентификаторы преобразований:

Имя	Номер	Определен в
RESERVED (зарезервировано)	0	
ENCR_DES_IV64	1	(RFC1827)
ENCR_DES	2	(RFC2405)
ENCR_3DES	3	(RFC2451)
ENCR_RC5	4	(RFC2451)
ENCR_IDEA	5	(RFC2451)
ENCR_CAST	6	(RFC2451)
ENCR_BLOWFISH	7	(RFC2451)
ENCR_3IDEA	8	(RFC2451)
ENCR_DES_IV32	9	
RESERVED	10	
ENCR_NULL	11	(RFC2410)
ENCR_AES_CBC	12	(RFC3602)
ENCR_AES_CTR	13	(RFC3664)

значения 14-1023 зарезервированы для IANA. Значения 1024-65535 предназначены для частного использования между взаимно согласными сторонами.

Для типа преобразования 2 (псевдослучайная функция) определены следующие идентификаторы преобразований:

Имя	Номер	Определен в
RESERVED (зарезервировано)	0	
PRF_HMAC_MD5	1	(RFC2104)
PRF_HMAC_SHA1	2	(RFC2104)
PRF_HMAC_TIGER	3	(RFC2104)
PRF_AES128_CBC	4	(RFC3664)

значения 5-1023 зарезервированы для IANA. Значения 1024-65535 предназначены для частного использования между взаимно согласными сторонами.

Для типа преобразования 3 (алгоритм целостности) определены следующие идентификаторы преобразований:

Имя	Номер	Определен в
NONE (отсутствует)	0	
AUTH_HMAC_MD5_96	1	(RFC2403)
AUTH_HMAC_SHA1_96	2	(RFC2404)
AUTH_DES_MAC	3	
AUTH_KPDK_MD5	4	(RFC1826)
AUTH_AES_XCBC_96	5	(RFC3566)

значения 6-1023 зарезервированы для IANA. Значения 1024-65535 предназначены для частного использования между взаимно согласными сторонами.

Для типа преобразования 4 (группа Диффи-Хеллмана) определены следующие идентификаторы преобразований:

Имя	Номер
NONE (отсутствует)	0
Определено в приложении B	1 - 2
RESERVED (зарезервировано)	3 - 4
Определено в [ADDGROUP]	5
Зарезервировано для IANA	6 - 13
Определено в [ADDGROUP]	14 - 18
Зарезервировано для IANA	19 - 1023
Для частного использования	1024-65535

Для типа преобразования 5 (расширенные порядковые номера) определены следующие идентификаторы преобразований:

Имя	Номер
No Extended Sequence Numbers (расширенные порядковые номера отсутствуют)	0
Extended Sequence Numbers (расширенные порядковые Номера)	1
RESERVED (зарезервировано)	2 - 65535

Если тип преобразования 5 не включается в предложение, то предполагается использование расширенных порядковых номеров.

3.3.3. Правомерные типы преобразований по протоколам

Количество и типы преобразований, которые следуют вместе с блоком данных SA, зависят от самого протокола в SA. Блок данных SA, предлагающий установление контекста безопасности, имеет следующие обязательные и необязательные типы преобразований. Реализация, соответствующая данной спецификации, должна (MUST) понимать все обязательные и необязательные типы для каждого протокола, который она поддерживает (хотя она не должна принимать предложения с неприемлемыми наборами). Предложение может (MAY) опускать необязательные типы, если единственным их значением, которое она будет принимать, является NONE (никакой).

Протокол	Обязательные типы	Необязательные типы
IKE	ENCR, PRF, INTEG, D-H	
ESP	ENCR	INTEG, D-H, ESN
AH	INTEG	D-H, ESN

3.3.4. Обязательные идентификаторы преобразований

Спецификация наборов, которые должны (MUST) поддерживаться и которые следует (SHOULD) поддерживать для интероперабельности из данного документа изъята, поскольку, вероятно, они меняются чаще, чем развивается данный документ.

Важным уроком, полученным от IKEv1, является то, что ни одна система не должна реализовывать только обязательные алгоритмы и предполагать, что они являются лучшим выбором для всех заказчиков. Например, во время написания данного документа многие разработчики реализаций IKEv1 начали переход на AES в режиме CBC для приложений VPN. Многие системы IPsec, базирующиеся на IKEv2 будут реализовывать AES, дополнительные группы Диффи-Хеллмана и дополнительные алгоритмы вычисления хэш-функций, и некоторые заказчики IPsec уже требуют эти алгоритмы в дополнение к перечисленным выше алгоритмам.

Вероятно IANA будет добавлять дополнительные преобразования в будущем, и некоторые пользователи могут захотеть использовать частные наборы, особенно для IKE, где реализации должны быть способными поддерживать различные параметры до ограничений определенного размера. Для поддержки этой цели все реализации IKEv2 должны (SHOULD) включать средство управления, которое допускает спецификацию (пользователем или системным администратором) параметров Диффи-Хеллмана для новых DH-групп (длины и значения генератора, модуля и показателя степени). Реализации должны (SHOULD) предоставлять интерфейс управления, с помощью которого эти параметры и связанные с ними идентификаторы преобразований могут вводиться (пользователем или системным администратором), чтобы разрешить согласование таких групп.

Все реализации IKEv2 должны (MUST) включать средство управления, которое позволяет пользователю или системному администратору определять наборы, которые приемлемы для использования с IKE. После получения блока данных с множеством идентификаторов преобразований реализация должна (MUST) сравнить переданные идентификаторы преобразований, чтобы проверить, что предлагаемый набор является приемлемым, базируясь на локальной политике. Реализации должны (MUST) отклонять предложения SA, которые не авторизованы этими средствами контроля наборов IKE. Заметим, что криптографические наборы, которые должны (MUST) быть реализованы, не нуждаются в конфигурировании как приемлемые для локальной политики.

3.3.5. Атрибуты преобразований

Каждое преобразование в блоке данных Контекст Безопасности может включать атрибуты, которые модифицируют или дополняют спецификацию преобразования. Эти атрибуты представляют собой пары тип/значение и определяются ниже. Например, если некоторый алгоритм шифрования имеет ключ переменной длины, то длина ключа, который должен использоваться, может быть специфицирована как атрибут. Атрибуты могут иметь значение фиксированной двухоктетной длины, или значение переменной длины. В последнем случае атрибут кодируется тройкой тип/длина/значение.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!A!       Attribute Type        !    AF=0  Attribute Length     !
!F!                             !    AF=1  Attribute Value      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                   AF=0  Attribute Value                       !
!                   AF=1  Not Transmitted                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

o Attribute Type – тип атрибута (2 октета) – Уникальный идентификатор для каждого типа атрибута (см. ниже).

  Наиболее значимый бит этого поля является битом формата атрибута (AF - Attribute Format). Он указывает, следуют ли атрибуты данных формату тип/длина/значение (TLV - Type/Length/Value) или укороченному формату тип/значение (TV - Type/Value). Если бит AF равен 0, то атрибуты данных представляются в форме TLV. Если бит AF равен 1, то атрибуты данных представляются в форме TV.

o Attribute Length – длина атрибута (2 октета) – Длина в октетах значения атрибута. Когда бит AF равен 1, значение атрибута имеет длину, равную только 2 октетам, и поле длины атрибута отсутствует.

o Attribute Value – значение атрибута (переменной длины) – Значение атрибута, связанное с типом атрибута. Если бит AF равен нулю (0), то это поле имеет переменную длину, определяемую полем Attribute Length. Если бит AF равен единице (1), то значение атрибута имеет длину 2 октета.

Заметим, что определен только один тип атрибута Key Length (длина ключа), и он имеет фиксированную длину. Спецификация кодирования атрибутов переменной длины включена в данный документ только для будущих расширений. В данном документе в качестве алгоритмов, принимающих атрибуты, определяются только алгоритмы шифрования, целостности и псевдослучайной функции, базирующиеся на AES, которые требуют одного атрибута, определяющего размер ключа.

Атрибуты, описанные как основные, не должны (MUST NOT) кодироваться с помощью кодирования переменной длины. Атрибуты переменной длины не должны (MUST NOT) кодироваться как основные, даже если их значение может поместиться в два октета. Примечание: Это изменение по сравнению с IKEv1, в котором для увеличения гибкости может применяться упрощенная функция составления сообщений, но существенно усложненная функция разбора сообщений.

Тип атрибута	Значение	Формат атрибута
RESERVED (зарезервировано)	0-13	
Key Length (длина ключа в битах)	14	TV
RESERVED (зарезервировано)	15-17	
RESERVED TO IANA (зарезервировано для IANA)	18-16383	

PRIVATE USE (для частного использования)   16384-32767

Значения 0-13 и 15-17 использовались в подобном контексте в IKEv1 и не должны присваиваться за исключением сопоставимых значений. Значения 18-16383 зарезервированы для IANA. Значения 16384-32767 предназначены для частного использования между взаимно согласными сторонами.

- Key Length - Длина ключа

  При использовании алгоритма шифрования, который имеет ключ переменной длины, этот атрибут определяет длину ключа в битах. (Должен (MUST) использоваться сетевой порядок байтов). Этот атрибут не должен (MUST NOT) использоваться, когда специфицированный алгоритм шифрования использует ключ фиксированной длины.

3.3.6. Согласование атрибутов

В процессе согласования контекста безопасности инициаторы представляют предложения ответчикам. Ответчики должны (MUST) выбрать из предложений один полный набор параметров (или отбросить все предложения, если ни одно не приемлемо). Если имеется несколько предложений, ответчик должен (MUST) выбрать один номер предложения и вернуть все структуры Предложение с этим номером предложения. Если имеется несколько преобразований одного и того же типа, ответчик должен (MUST) выбрать одно преобразование. Все атрибуты выбранного преобразования должны (MUST) быть возвращены без изменений. Инициатор обмена должен (MUST) проверить, что принятое предложение согласуется с одним из его предложений, и если не согласуется, то такой ответ должен (MUST) быть отброшен.

Согласование групп Диффи-Хеллмана представляет некоторые конкретные проблемы. Предложения SA включают в одно и то же сообщение предлагаемые атрибуты и открытое число Диффи-Хеллмана (KE). Если в начальном обмене инициатор предлагает использовать одну из нескольких групп Диффи-Хеллмана, он должен (SHOULD) выбрать ту, которую вероятнее всего примет ответчик, и включить KE, соответствующий этой группе. Если выяснится, что догадка была ошибочной, ответчик укажет правильную группу в ответе и инициатор должен (SHOULD) выбрать элемент этой группы для своего значения KE при повторной посылке первого сообщения. Однако он должен (SHOULD) продолжать предлагать свой полный набор поддерживаемых групп, чтобы предотвратить атаку типа "человек посередине", понижающую степень защиты.

Замечание по реализации:

> Некоторые согласуемые атрибуты могут иметь диапазоны или иметь несколько приемлемых значений. К таким атрибутам относится длина ключа симметричного шифра с переменной длиной ключа. Для обеспечения интероперабельности в будущем, а также для поддержки независимого обновления оконечных точек разработчики реализаций данного протокола должны (SHOULD) принимать значения, которые, по их мнению, обеспечивают большую безопасность. Например, если партнер конфигурируется для приема шифра переменной длины с длиной ключа X бит и предлагается этот шифр с большей длиной ключа, то реализация должна (SHOULD) принять это предложение, если она поддерживает использование более длинного ключа.

Поддержка этой возможности позволяет реализации выразить концепцию "не менее" определенного уровня безопасности - "a key length of _at least_ X bits for cipher Y" (длина ключа не менее X бит для шифра Y).

3.4. Блок данных Обмен Ключами

Блок данных Обмен Ключами (Key Exchange Payload), обозначаемый в этом меморандуме KE, используется для обмена открытыми числами Диффи-Хеллмана как часть обмена ключами Диффи-Хеллмана. Блок данных Обмен Ключами включает общий заголовок блоков данных IKE, за которым следует само открытое значение Диффи-Хеллмана.

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!          DH Group #           !           RESERVED            !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Key Exchange Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 10. Формат блока данных Обмен Ключами

Блок данных Обмен Ключами создается путем копирования открытого значения Диффи-Хеллмана в поле "Key Exchange Data" этого блока данных. Длина открытого значения Диффи-Хеллмана должна (MUST) быть равна длине простого модуля, по которому выполнялось возведение в степень, при этом, если необходимо, значение дополняется спереди нулевыми битами.

Поле DH Group # указывает группу Диффи-Хеллмана, в которой вычислялись данные обмена ключами - Key Exchange Data (см. подраздел 3.3.2). Если выбранное предложение использует другую группу Диффи-Хеллмана, то сообщение должно (MUST) быть отклонено, и в ответе посылается блок данных Уведомление типа INVALID_KE_PAYLOAD (неправильный блок данных KE).

Тип блока данных Обмен Ключами имеет значение тридцать четыре (34).

3.5. Блоки данных Идентификация

Блоки данных Идентификация (Identification Payloads), обозначаемые в данном меморандуме IDi и IDr, позволяют партнерам объявить друг другу свои идентификаторы. Этот идентификатор может использоваться для поиска в политике, но необязательно должен совпадать с чем-либо в блоке данных CERT; оба поля могут использоваться реализацией для принятия решения по контролю доступа.

Примечание: В IKEv1 при передаче в каждом направлении использовались два блока данных Идентификация (ID) для размещения информации о селекторах трафика (Traffic Selector) для данных, передаваемых через SA. В IKEv2 эта информация передается в блоках данных Селекторы Трафика (TS) (см. подраздел 3.13).

Блок данных Идентификация состоит из общего заголовка блоков данных IKE, за которым следуют поля идентификаторов следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !                 RESERVED                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 11. Формат блока данных Идентификация.

- o ID Type – тип идентификатора (1 октет) – Определяет тип используемой идентификации.

- o RESERVED – зарезервировано – должно (MUST) посылаться нулевым и должно (MUST) игнорироваться при получении.

- o Identification Data – данные идентификации (переменной длины) – Значение, указанное типом идентификации. Длина данных идентификации вычисляется исходя из длины блока данных Идентификация, указанной в заголовке.

Тип блока данных Идентификация имеет значение тридцать пять (35) для IDi и тридцать шесть (36)для IDr.

В следующей таблице перечислены значения, присвоенные полю Identification Type (тип идентификации), за которыми следует описание данных идентификации:

```
Тип идентификатора              Значение
------------------              --------
RESERVED (зарезервировано)         0
ID_IPV4_ADDR                       1
      Одиночный IPv4-адрес длиной четыре (4) октета
ID_FQDN                            2
      Строка полностью квалифицированного доменного имени. Примером
      ID_FQDN является "example.com". Строка не должна (MUST not)
      содержать никаких признаков конца (например, NULL, CR, и т.д.).
ID_RFC822_ADDR                     3
      Строка полностью квалифицированного адреса электронной почты в
      соответствии с RFC822. Примером ID_RFC822_ADDR является
      "jsmith@example.com". Строка не должна (MUST not) содержать никаких
      признаков конца.
Reserved to IANA                   4
      Зарезервировано для IANA
ID_IPV6_ADDR                       5
      Одиночный IPv6-адрес длиной шестнадцать (16) октетов
Reserved to IANA                   6 - 8
      Зарезервировано для IANA
ID_DER_ASN1_DN                     9
      Двоичная кодировка DER ASN.1 различительного имени (Distinguished
      Name) X.500 [X.501].
ID_DER_ASN1_GN                     10
      Двоичная кодировка DER ASN.1 общего имени (GeneralName) X.500
      [X.509].
ID_KEY_ID                          11
      Непрозрачный поток октетов, который может использоваться для
      передачи конкретной информации, необходимой для определенных
      патентованных типов идентификации.
Reserved to IANA                   12-200
      Зарезервировано для IANA
Reserved for private use           201-255
      Зарезервировано для частного использования
```

Две реализации смогут взаимодействовать, только если каждая из них может генерировать тип идентификатора, приемлемый для другой стороны. Для обеспечения максимальной интероперабельности, реализации должны (MUST) иметь возможность конфигурирования для посылки по крайней мере одного типа идентификатора ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR или ID_KEY_ID и должны (MUST) иметь возможность конфигурирования для приема всех этих типов идентификации. Реализации должны (SHOULD) быть способными генерировать и принимать все эти типы идентификации. Реализации, обеспечивающие работу с IPv6, должны (MUST) дополнительно иметь возможность конфигурирования для приема ID_IPV6_ADDR. Реализации, работающие только по протоколу IPv6, могут (MAY) конфигурироваться только для посылки ID_IPV6_ADDR.

3.6. Блок данных Сертификат

Блок данных Сертификат (Certificate Payload), обозначаемый в данной спецификации CERT, обеспечивает средство транспортировки с помощью IKE сертификатов или другой информации, связанной с аутентификацией. Блоки данных Сертификат должны (SHOULD) включаться в обмен, если отправителю доступны сертификаты, за исключением случаев, когда партнер указал способность извлечения этой информации из какого-то другого места с помощью блока данных Уведомление (Notify) типа HTTP_CERT_LOOKUP_SUPPORTED (поддерживается HTTP-поиск сертификата). Заметим, что термин "блок данных Сертификат" слегка вводит в заблуждение, поскольку не все механизмы аутентификации используют сертификаты, и в этом блоке данных могут передаваться данные, отличные от сертификатов.

Блок данных Сертификат определяется следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                       Certificate Data                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 12. Формат блока данных Сертификат

- Certificate Encoding – кодировка сертификата (1 октет) – Это поле указывает тип сертификата или связанной с сертификатом информации, содержащейся в поле Certificate Data.

Кодирование сертификата	Значение
RESERVED (зарезервировано)	0
PKCS #7 wrapped X.509 certificate (Сертификат X.509, в обертке PKCS #7)	1
PGP Certificate (Сертификат PGP)	2
DNS Signed Key (Ключ, подписанный DNS)	3
X.509 Certificate - Signature (Сертификат X.509 – Подпись)	4
Kerberos Token (Маркер Kerberos)	6
Certificate Revocation List (CRL) (Список аннулированных сертификатов)	7
Authority Revocation List (ARL) (Список аннулированных центров сертификации)	8
SPKI Certificate (Сертификат SPKI)	9
X.509 Certificate - Attribute (Сертификат X.509 – Атрибут)	10
Raw RSA Key (Предварительный ключ RSA)	11
Hash and URL of X.509 certificate (Хэш-значение и URL сертификата X.509)	12
Hash and URL of X.509 bundle (Хэш-значение и URL связки X.509)	13
RESERVED to IANA (зарезервировано для IANA)	14 - 200
PRIVATE USE (частного пользования)	201 – 255

- Certificate Data – данные сертификата (переменной длины) – Фактическое кодирование данных сертификации. Тип сертификата указывается полем Certificate Encoding.

Тип блока данных Сертификат имеет значение тридцать семь (37).

Конкретный синтаксис для некоторых перечисленных выше кодов типов сертификатов в данном документе не определяется. В данном документе определяется синтаксис следующих типов:

X.509 Certificate - Signature (Сертификат – Подпись X.509) (4) содержит сертификат X.509 в кодировке DER, открытый ключ которого используется для подтверждения блока данных AUTH отправителя.

Certificate Revocation List (Список аннулированных сертификатов) (7) содержит список аннулированных сертификатов X.509 в кодировке DER.

Raw RSA Key (Предварительный ключ RSA) (11) содержит ключ RSA в кодировке PKCS #1.

Кодировки хэш-значения и URL (12-13) позволяют IKE-сообщениям оставаться короткими, путем замены длинных структур данных сертификата 20-октетным значением хэш-функции SHA-1 (см. [SHA]) от замененного значения, за которым следует универсальный локатор ресурса (URL) переменной длины, который сам преобразуется в структуру данных в кодировке DER. Это повышает эффективность, когда оконечные точки имеют кэшированные данные сертификатов, и делает IKE менее восприимчивым к атакам на доступность, которые легче организовать в случае достаточно длинных сообщений IKE, требующих IP-фрагментации [KPS03].

Используйте следующее ASN.1-определение для связки сертификатов X.509:

```
CertBundle
  { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-cert-bundle(34) }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS
  Certificate, CertificateList
  FROM PKIX1Explicit88
     { iso(1) identified-organization(3) dod(6)
       internet(1) security(5) mechanisms(5) pkix(7)
       id-mod(0) id-pkix1-explicit(18) } ;

CertificateOrCRL ::= CHOICE {
  cert [0] Certificate,
  crl  [1] CertificateList }

CertificateBundle ::= SEQUENCE OF CertificateOrCRL

END
```

Для поддержки аутентификации реализации должны (MUST) иметь возможность конфигурирования посылки и приема до четырех сертификатов X.509, а также должны (MUST) иметь возможность конфигурирования посылки и приема первых двух форматов Hash and URL (с HTTP URL). Реализации должны (SHOULD) иметь возможность конфигурирования для посылки и приема предварительных ключей RSA. Если посылается несколько сертификатов, первый сертификат должен (MUST) содержать открытый ключ, используемый для подписи блока данных AUTH. Другие сертификаты могут посылаться в любом порядке.

3.7. Блок данных Запрос Сертификата

Блок данных Запрос Сертификата (Certificate Request Payload), обозначаемый в данном меморандуме CERTREQ, обеспечивает средство запроса с помощью IKE предпочтительных сертификатов, и может появляться в ответе IKE_INIT_SA и/или в запросе IKE_AUTH. Блоки данных Запрос Сертификата могут (MAY) включаться в обмен, когда отправителю необходимо получить сертификат получателя. Если доверительными являются несколько центром сертификации CA (Certification Authority), а кодирование cert не допускает списка, то должны (SHOULD) передаваться несколько блоков данных Запрос Сертификата.

Блок данных Запрос Сертификата определяется следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                    Certification Authority                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 13. Формат блока данных Запрос Сертификата.

- Certificate Encoding – кодировка сертификата (1 октет) – Содержит кодировку типа или формата запрашиваемого сертификата. Значения перечислены в подразделе 3.6.

- Certification Authority – центр сертификации (переменной длины) – Содержит кодировку приемлемого центра сертификации для типа запрашиваемого сертификата.

Тип блока данных Запрос Сертификата имеет значение тридцать восемь (38).

Поле Certificate Encoding имеет те же самые значения, что и значения, определенные в подразделе 3.6. Поле Certification Authority содержит указатель доверенных центров для сертификатов данного типа. Значение Certification Authority представляет собой конкатенированный список значений хэш-функции SHA-1 от открытых ключей доверенных CA. Каждое значение кодируется в виде значения хэш-функции SHA-1 от элемента Subject Public Key Info (см. пп. 4.1.2.7 [RFC3280]) от каждого сертификата Trust Anchor. Двадцатиоктетные значения хэш-функции конкатенируются и включаются в поле Certification Authority без какого-либо дополнительного форматирования.

Заметим, что термин "запрос сертификата" (Certificate Request) слегка вводит в заблуждение в том, что в блоке данных "Certificate" определяются значения, отличные от сертификатов, и запрос таких значений может находиться в блоке данных Запрос Сертификата. В данном документе синтаксис блока данных Запрос Сертификата для таких случаев не определяется.

Блок данных Запрос Сертификата обрабатывается путем инспектирования поля "Cert Encoding", чтобы определить, имеет ли обработчик какие-либо сертификаты данного типа. Если это так, то инспектируется поле "Certification Authority", чтобы определить, имеет ли обработчик какие-либо сертификаты, которые могут быть признаны действительными вплоть до сертификата одного из специфицированных центров сертификации. Это может быть цепочка сертификатов.

Если существует сертификат оконечной сущности, который удовлетворяет критериям, указанным в CERTREQ, то сертификат или цепочка сертификатов должны (SHOULD) быть посланы назад запросчику сертификата, если:

- получатель CERTREQ конфигурируется для использования аутентификации на основе сертификатов,

- разрешается посылать блок данных CERT,

- имеет соответствующую политику доверия CA, контролирующую текущее согласование, и

- имеет по крайней мере одну приемлемую по времени и подходящую для использования цепочку сертификатов оконечной сущности к CA, представленному в CERTREQ.

Проверка аннулирования сертификатов должна рассматриваться в процессе обработки цепочки, используемой для выбора сертификата. Заметим, что даже если оба партнера конфигурируются для использования двух разных CA, соответствующей логикой выбора должны поддерживаться взаимоотношения перекрестной сертификации. Цель заключается не в предотвращении обмена информацией, а в строгом соблюдении правил выбора сертификата на основе CERTREQ, когда отправителем может быть выбран альтернативный сертификат, который все еще позволит получателю успешно признать его действительным и поверить ему, хотя доверие передается посредством перекрестной сертификации, списков аннулированных сертификатов CRL или других конфигурируемых внепротокольных (out-of-band) средств. Таким образом, обработка CERTREQ должна рассматриваться как предложение выбора сертификата, а не приказ. Если сертификаты отсутствуют, то CERTREQ игнорируется. Для протокола это не является условием ошибки. Могут быть ситуации, когда существует предпочтительный CA, посланный в CERTREQ, но может оказаться приемлемым и альтернативный CA (возможно после консультации с человеком-оператором).

3.8. Блок данных Аутентификация

Блок данных Аутентификация (Authentication Payload), обозначаемый в данном меморандуме AUTH, содержит данные, используемые с целью аутентификации. Как определено ниже, синтаксис данных аутентификации меняется в соответствии с методом аутентификации (Auth Method).

Блок данных Аутентификация определяется следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Auth Method   !                RESERVED                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                      Authentication Data                      ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 14. Формат блока данных Аутентификация.

- o Auth Method – метод аутентификации (1 октет) – Определяет используемый метод аутентификации. Определенны следующие значения:

  RSA Digital Signature – цифровая подпись RSA (1) – Вычисляется так, как определено в подразделе 2.15 с использованием секретного ключа RSA от дополненного хэш-значения PKCS#1 (см. [RSA] и [PKCS1]).

  Shared Key Message Integrity Code – код целостности сообщения, управляемый общим ключом (2) – Вычисляется так, как определено в подразделе 2.15 с использованием общего ключа, ассоциированного с идентификатором из блока данных ID, и согласованной функции prf.

  DSS Digital Signature – цифровая подпись DSS (3) – Вычисляется так, как определено в подразделе 2.15 с использованием секретного ключа DSS (см. [DSS]) от значения хэш-функции SHA-1.

  Значения 0 и 4-200 зарезервированы для IANA. Значения 201-255 доступны для частного использования.

- o Authentication Data – данные аутентификации (переменной длины) – см. подраздел 2.15.

Тип блока данных Аутентификация имеет значение тридцать девять (39).

3.9. Блок данных Одноразовый Номер

Блок данных Одноразовый Номер (Nonce Payload), обозначаемый в данном меморандуме Ni и Nr для одноразового номера инициатора и ответчика, соответственно, содержит случайные данные, используемые для гарантии работоспособности в процессе обмена, а также для защиты от атак повторного воспроизведения.

Блок данных Одноразовый Номер определяется следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                            Nonce Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 15. Формат блока данных Одноразовый Номер.

- o Nonce Data – данные одноразового номера (переменной длины) – Содержит случайные данные, генерируемые передающей сущностью.

Тип блока данных Одноразовый Номер имеет значение сорок (40).

Длина одноразового номера должна (MUST) находиться в диапазоне от 16 до 256 октетов включительно. Значения одноразовых номеров не должны (MUST NOT) повторно использоваться.

3.10. Блок данных Уведомление

Блок данных Уведомление (Notify Payload), обозначаемый в данном документе N, используется для передачи партнеру IKE информационных данных, например, условий ошибок и переходов состояний. Блок данных Уведомление может появляться в сообщении ответа (обычно указывая причину отклонения запроса), в информационном (INFORMATIONAL) обмене (для сообщения об ошибке не в запросе IKE) или в любом другом сообщении для указания возможностей отправителя или для изменения смысла запроса.

Блок данных Уведомление определяется следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 16. Формат блока данных Уведомление.

- o Protocol ID – идентификатор протокола - (1 октет) – Если данное уведомление касается существующего SA, то это поле указывает тип этого SA. Для уведомлений IKE_SA это поле должно (MUST) быть равно единице (1). Для уведомлений, касающихся IPsec SA, это поле должно (MUST) содержать либо (2) для указания AH, либо (3) для указания ESP. Для уведомлений, которые не связаны с существующим SA, это поле должно (MUST) посылаться нулевым и должно (MUST) игнорироваться при приеме. Все другие значения этого поля зарезервированы для будущего назначения IANA.

- o SPI Size – длина SPI (1 октет) – Длина SPI в октетах, которая определяется идентификатором протокола IPsec, или ноль, если SPI не применим. Для уведомления, касающегося IKE_SA, поле SPI Size должно (MUST) быть нулевым.

- o Notify Message Type – тип сообщения уведомления (2 октета) – Определяет тип сообщения уведомления.

- o SPI - Security Parameter Index (переменной длины) – индекс параметров безопасности.

- o Notification Data – данные уведомления - (переменной длины) – Информационные данные или данные об ошибках, передаваемые в дополнение к типу сообщения уведомления. Значения для этого поля зависят от типа сообщения уведомления (см. ниже).

Тип блока данных Уведомление имеет значение сорок один (41).

3.10.1. Типы сообщений Уведомление

Уведомительная информация может содержать сообщение об ошибке, определяющим, почему не может быть установлен SA. Это могут быть также данные о состоянии, которые процесс, управляющий базой данных SA, хочет сообщить процессу-партнеру. В нижеприведенной таблице перечислены сообщения-уведомления и их соответствующие значения. По сравнению с IKEv1 количество различных состояний отказа было значительно уменьшено как для упрощения, так и для того, чтобы избежать выдачи конфигурационной информации злоумышленникам, осуществляющим зондаж узла.

Типы в диапазоне 0 – 16383 предназначены для сообщений об ошибках. Реализация, получающая блок данных Уведомление с одним из таких типов, которые она не распознает в ответе, должна (MUST) предположить, что на соответствующий запрос пришел полный отказ. Нераспознаваемые типы ошибок в запросе и типы состояний в запросе или ответе должны (MUST) игнорироваться, за исключением того, что они должны (SHOULD) записываться в журнал.

Блоки данных Уведомление с типами состояния могут (MAY) быть добавлены к любому сообщению и должны (MUST) игнорироваться, если они не распознаются. Они предназначены для указания возможностей и, как часть согласования SA, используются для согласования не криптографических параметров.

```
Сообщения уведомления – Типы ошибок                      Значение
-----------------------------------                      --------
RESERVED (зарезервировано)                               0

UNSUPPORTED_CRITICAL_PAYLOAD                             1
(Неподдерживаемый критический блок данных)
   Посылается, если блок данных имеет установленный бит "critical" и тип
   блока данных не распознается. Поле Notification Data содержит один
   октет типа блока данных.

INVALID_IKE_SPI                                          4
(Неправильный IKE_SPI)
   Указывает на то, что сообщение IKE было получено с нераспознаваемым SPI
   назначения. Обычно это указывает на то, что получатель был перезапущен
   и забыл о наличии IKE_SA.
```

INVALID_MAJOR_VERSION 5
(Неправильная старшая часть номера версии)
Указывает на то, что получатель не может обрабатывать версию IKE, указанную в заголовке. В заголовке ответа будет находиться ближайший номер версии, которую может поддерживать получатель.

INVALID_SYNTAX 7
(Неправильный синтаксис)
Указывает на то, что полученное сообщение IKE было неправильным из-за того, что некоторый тип, длина или значение вышли из диапазона, или из-за того, что запрос был отклонен по причинам, связанным с политикой. Чтобы избежать атаки на доступность, использующей подложные сообщения, этот тип ошибки может возвращаться только для зашифрованного пакета и только в зашифрованном пакете, если идентификатор сообщения (message ID) и криптографическая контрольная сумма были правильными. Чтобы избежать утечки информации кому-то, кто осуществляет зондаж узла, этот тип ошибки должен посылаться (MUST) в ответ на какую-либо ошибку, не покрытую одним из других типов ошибок. Чтобы помочь в отладке, более подробная информация об ошибке должна (SHOULD) выводиться на консоль или записываться в журнал.

INVALID_MESSAGE_ID 9
(Неправильный идентификатор сообщения)
Посылается, когда принимается идентификатор сообщения IKE, находящийся за пределами поддерживаемого окна. Это уведомление не должно (MUST NOT) посылаться в ответе; неправильный запрос не должен (MUST NOT) подтверждаться. Вместо этого, другая сторона должна информироваться путем инициации информационного (INFORMATIONAL) обмена с данными уведомления, содержащими четыре октета неправильного идентификатора сообщения. Посылка такого уведомления не является обязательной, и уведомления такого типа должны (MUST) ограничиваться по скорости.

INVALID_SPI 11
(Неправильный SPI)
Может (MAY) посылаться в информационном (INFORMATIONAL) обмене IKE, когда узел получает ESP- или AH-пакет с неправильным SPI. Данные уведомления содержат SPI неправильного пакета. Обычно это указывает на то, что узел перезагрузился и забыл SA. Если это информационное сообщение посылается вне контекста некоторого IKE_SA, то оно должно использоваться получателем только как "намек" на то, что что-то не правильно (поскольку его можно легко подделать).

NO_PROPOSAL_CHOSEN 14
(Предложение не выбрано)
Ни один из предложенных криптографических наборов не приемлем.

INVALID_KE_PAYLOAD 17
(Неправильный блок данных KE)
Поле D-H Group # в блоке данных KE не является номером группы, выбранной ответчиком для данного обмена. Имеются два октета данных, связанных с этим уведомлением: принятый номер группы Диффи-Хеллмана (D-H Group #) в сетевом порядке бит.

AUTHENTICATION_FAILED 24
(Аутентификация не прошла)
Посылается в ответе на сообщение IKE_AUTH, когда по каким-то причинам аутентификация не прошла. Какие-либо связанные с ним данные отсутствуют.

SINGLE_PAIR_REQUIRED 34
(Требуется одна пара)
Эта ошибка указывает на то, что запрос CREATE_CHILD_SA является неприемлемым, поскольку ее отправитель хочет принимать селекторы трафика, определяя только одну пару адресов. Предполагается, что

запросчик ответит запросом SA только для конкретного трафика, который он пытается переслать.

NO_ADDITIONAL_SAS 35
(Дополнительные SA не поддерживаются)
Эта ошибка указывает на то, что запрос CREATE_CHILD_SA является неприемлемым, поскольку ответчик не хочет принимать дополнительные CHILD_SA по этому IKE_SA. Некоторые минимальные реализации могут принять только одну установку CHILD_SA в контексте начального обмена IKE и отвергают какие-либо последующие попытки установить дополнительные контексты безопасности.

INTERNAL_ADDRESS_FAILURE 36
(Ошибка внутреннего адреса)
Указывает ошибку присваивания внутреннего адреса (т.е. INTERNAL_IP4_ADDRESS или INTERNAL_IP6_ADDRESS) во время обработки ответчиком блока данных Конфигурирование (Configuration Payload). Если эта ошибка генерируется в рамках обмена IKE_AUTH, то никакого CHILD_SA не будет открыто.

FAILED_CP_REQUIRED 37
(Требуется отсутствующий блок данных CP)
Посылается ответчиком в случае, когда ожидался, но не был получен блок данных CP(CFG_REQUEST), и, таким образом, имеется конфликт с локально сконфигурированной политикой. Связанные с ним данные отсутствуют.

TS_UNACCEPTABLE 38
(Неприемлемый TS)
Указывает на то, что ни один из адресов/протоколов/портов в представленных селекторах трафика не является приемлемым.

INVALID_SELECTORS 39
(Неправильные селекторы)
Может (MAY) посылаться в информационном (INFORMATIONAL) обмене IKE, когда узел получает ESP- или AH-пакет, селекторы которого не соответствуют селекторам SA, по которому он был доставлен (и которые привели к отбрасыванию пакета). Поле Notification Data содержит начало неправильного пакета (как в сообщениях ICMP), а поле SPI уведомления устанавливается так, чтобы оно соответствовало IPsec SA.

RESERVED TO IANA - Error types 40 - 8191
(Зарезервировано для IANA – Типы ошибок)

Private Use - Errors 8192 - 16383
(Частного пользования – Ошибки)

Сообщения уведомления – Типы состояния	Значение

INITIAL_CONTACT 16384
(Начальный контакт)
Это уведомление утверждает, что данный IKE_SA является единственным IKE_SA между аутентифицированными идентификаторами, который в текущий момент времени является активным. Оно может (MAY) посылаться, когда IKE_SA устанавливается после аварии, и получатель может (MAY) использовать эту информацию для удаления без ожидания тайм-аута любых других IKE_SA, которые он имеет с тем же самым идентификатором. Это уведомление не должно (MUST NOT) посылаться сущностью, которая может быть реплицирована (например, в случае роуминга мандатов пользователя, при котором пользователю позволено подсоединяться к корпоративному межсетевому экрану из двух удаленных систем одновременно).

SET_WINDOW_SIZE 16385

(Установить размер окна)
Это уведомление утверждает, что посылающая оконечная точка способна хранить состояние нескольких исходящих обменов, позволяя получателю послать несколько запросов до получения ответа на первый запрос. Данные, связанные с уведомлением SET_WINDOW_SIZE должны (MUST) иметь длину в 4 октета, содержать количество сообщений, которые отправитель обещает хранить, и располагаться в сетевом порядке бит. Размер окна всегда равен единице до тех пор, пока не завершатся начальные обмены.

ADDITIONAL_TS_POSSIBLE 16386
(Возможны дополнительные TS)
Это уведомление объявляет, что пославшая его оконечная точка сузила предлагаемые селекторы трафика, и что другие селекторы трафика будут также приемлемыми, но только в отдельном SA (см. подраздел 2.9). С этим типом уведомления не связано никаких данных. Оно может посылаться только в качестве дополнительного блока данных в сообщении, включающем принятые селекторы трафика.

IPCOMP_SUPPORTED 16387
(IPCOMP поддерживается)
Это уведомление может включаться только в сообщение, содержащее блок данных SA, согласующий CHILD_SA, и указывает желание его отправителя использовать IPComp на этом SA. Данные, связанные с этим уведомлением, включают двухоктетный IPComp CPI, за которым следует один октет идентификатора преобразования, за которым могут следовать необязательные атрибуты, длина которых и формат определяются этим идентификатором преобразования. Сообщение, предлагающее SA, может содержать несколько уведомлений IPCOMP_SUPPORTED для указания нескольких поддерживаемых алгоритмов. Сообщение, принимающее SA, может содержать только одно уведомление.

В настоящее время определены следующие идентификаторы преобразований:

Имя	Номер	Определено в
RESERVED	0	
IPCOMP_OUI	1	
IPCOMP_DEFLATE	2	RFC 2394
IPCOMP_LZS	3	RFC 2395
IPCOMP_LZJH	4	RFC 3051

Значения 5-240 зарезервированы для IANA. Значения 241-255 предназначены для частного использования между взаимно согласными сторонами.

NAT_DETECTION_SOURCE_IP 16388
(Определение нахождения IP-адреса источника за NAT)
Это уведомление используется его получателем для определения того, находится ли источник за устройством NAT. Данные, связанные с этим уведомлением, содержат дайджест, вычисленный с помощью алгоритма SHA-1, от индексов параметров безопасности (в том порядке, в котором они появляются в заголовке), IP-адреса и порта, с которых этот пакет посылался. В сообщении может (MAY) быть несколько блоков данных этого типа, если отправитель не знает, какое из нескольких подключений к сети будет использоваться для посылки пакета. Получатель этого уведомления может (MAY) сравнить представленное значение со значением хэш-функции SHA-1 от индексов параметров безопасности, IP-адреса и порта источника, и в случае несоответствия он должен (SHOULD) разрешить пересечение NAT (см. подраздел 2.23). Альтернативно, он может (MAY) отклонить попытку соединения, если пересечение NAT не поддерживается.

NAT_DETECTION_DESTINATION_IP 16389
(Определение нахождения IP-адреса назначения за NAT)

Это уведомление используется его получателем для определения того, находится ли он за устройством NAT. Данные, связанные с этим уведомлением, представляют собой дайджест, вычисленный с помощью алгоритма SHA-1, от индексов параметров безопасности (в том порядке, в котором они появляются в заголовке), IP-адреса и порта, на которые этот пакет посылался. Получатель этого уведомления может (MAY) сравнить представленное значение со значением хэш-функции SHA-1 от индексов параметров безопасности, IP-адреса и порта назначения, и в случае несоответствия он должен (SHOULD) вызвать пересечение NAT (см. подраздел 2.23). Отсутствие соответствия означает, что этот конец соединения находится за NAT и должен (SHOULD) начать посылку пакетов подтверждения работоспособности (keepalive), как это определено в [Hutt05]. Альтернативно, он может (MAY) отклонить попытку соединения, если пересечение NAT не поддерживается.

COOKIE 16390
(Идентифицирующая цепочка)
Это уведомление может (MAY) включаться в ответ IKE_SA_INIT. Оно указывает на то, что запрос должен быть послан повторно, и содержать в первом блоке данных копию данного уведомления. Это уведомление должно (MUST) включаться в повторную попытку запроса IKE_SA_INIT, если уведомление COOKIE было включено в начальный ответ. Данные, связанные с этим уведомлением, должны (MUST) быть длиною от 1 до 64 октетов (включительно).

USE_TRANSPORT_MODE 16391
(Использовать транспортный режим)
Это уведомление может (MAY) включаться в сообщение запроса, который включает также блок данных SA, запрашивающий CHILD_SA. Оно требует, чтобы CHILD_SA для создаваемого контекста безопасности использовал транспортный, а не туннельный режим. Если такой запрос принимается, то ответ также должен (MUST) включать уведомление USE_TRANSPORT_MODE. Если ответчик отклоняет запрос, то CHILD_SA будет установлен в туннельном режиме. Если это неприемлемо для инициатора, инициатор должен (MUST) уничтожить SA. Примечание: Все CHILD_SA будут использовать туннельный режим, за исключением случая, когда для согласования транспортного режима используется эта опция.

Примечание: Изменения, связанные с разборкой (decapsulation) ECN, специфицированные в [RFC4301], должны (MUST) выполняться для каждого созданного IKEv2 контекста безопасности туннельного режима.

HTTP_CERT_LOOKUP_SUPPORTED 16392
(Поддерживается HTTP-поиск сертификата)
Это уведомление может (MAY) включаться в любое сообщение, которое может содержать блок данных CERTREQ, и указывает на то, что отправитель способен осуществлять поиск сертификатов, основываясь на URL, базирующемся на HTTP (и поэтому, вероятно, предпочтет получать спецификации сертификатов в этом формате).

REKEY_SA 16393
(Переустановка SA)
Это уведомление должно (MUST) включаться в обмен CREATE_CHILD_SA, если целью обмена является замена существующего ESP SA или AH SA. Поле SPI определяет переустанавливаемый SA. Какие-либо данные отсутствуют.

ESP_TFC_PADDING_NOT_SUPPORTED 16394
(Заполнение ESP_TFC не поддерживается)
Это уведомление объявляет о том, что посылающая оконечная точка не будет принимать пакеты, которые содержат заполнитель Конфиденциальность потока трафика (TFC – Traffic Flow Confidentiality).

NON_FIRST_FRAGMENTS_ALSO 16395
(Также никаких первых фрагментов)

Используется для управления фрагментацией. См. [RFC4301] для пояснения.

RESERVED TO IANA - STATUS TYPES 16396 - 40959
(Зарезервировано для IANA – типы состояний)

Private Use - STATUS TYPES 40960 - 65535
(Для частного использования – типы состояний)

3.11. Блок данных Удаление

Блок данных Удаление (Delete Payload), обозначаемый в данном меморандуме D, содержит зависящий от протокола идентификатор конкретного контекста безопасности, который отправитель удалил из своей базы данных контекстов безопасности, и который, поэтому, больше не является действительным. На рис. 17 представлен формат уведомления Удаление. В блоке данных Удаление можно посылать несколько SPI, но каждый SPI должен (MUST) соответствовать одному и тому же протоколу. Однако в один информационный (INFORMATIONAL) обмен разрешается включать несколько блоков данных Удаление, когда в каждом отдельном блоке данных Удаление перечислены SPI для различных протоколов.

Удаление IKE_SA указывается идентификатором протокола, равным 1 (IKE), а не индексами параметров безопасности. Удаление CHILD_SA, например, ESP или AH, будет содержать идентификатор (IPsec protocol ID) этого протокола (2 для AH, 3 для ESP), а SPI будет представлять собой SPI, который посылающая оконечная точка предполагает во входящих ESP- или AH-пакетах.

Блок данных Удаление определяется следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Protocol ID   !   SPI Size    !           # of SPIs           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~               Security Parameter Index(es) (SPI)              ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 17. Формат блока данных Удаление.

- o Protocol ID – идентификатор протокола (1 октет) – Должен быть равен 1 для IKE_SA, 2 для AH, или 3 для ESP.

- o SPI Size – длина SPI (1 октет) – Длина SPI в октетах, которая определяется идентификатором протокола. Она должна (MUST) быть равна нулю для IKE (SPI находится в заголовке сообщения) или четырем для AH и ESP.

- o # of SPIs – количество SPI, (2 октета) – Количество SPI, содержащихся в блоке данных Удаление. Размер каждого SPI определяется полем SPI Size.

- o Security Parameter Index(es) – индекс (индексы) параметров безопасности (переменной длины) – Указывает конкретный контекст безопасности (конкретные контексты безопасности), подлежащий (подлежащие) удалению. Длина этого поля определяется полями SPI Size и # of SPIs.

Тип блока данных Удаление имеет значение сорок два (42).

3.12. Блок данных Идентификатор Поставщика

Блок данных Идентификатор Поставщика (Vendor ID Payload) содержит константу, определяемую поставщиком. Эта константа используется поставщиками для указания и распознавания удаленных копий своих реализаций. Этот механизм позволяет поставщику проводить эксперименты с новыми возможностями при поддержке обратной совместимости.

Блок данных Идентификатор Поставщика может (MAY) извещать о том, что отправитель способен принимать определенные расширения протокола, или может (MAY) просто идентифицировать реализацию как вспомогательное средство при отладке. Блок данных Идентификатор Поставщика не должен (MUST NOT) менять интерпретацию какой-либо информации, определенной в данной спецификации (т.е. бит critical должен (MUST) быть установлен в ноль). Могут (MAY) посылаться несколько блоков данных Идентификатор Поставщика. От реализаций вообще не требуется (NOT REQUIRED) посылать какие-либо блоки данных Идентификатор Поставщика.

Блок данных Идентификатор Поставщика может посылаться как часть любого сообщения. Получение известного блока данных Идентификатор Поставщика позволяет реализации использовать номера частного пользования (Private USE), описанные в данном меморандуме - частные блоки данных, частные обмены, частные уведомления и т.д. Неизвестные идентификаторы поставщика должны (MUST) игнорироваться.

Авторы черновых вариантов проектов Internet (Internet-Drafts), которые хотят расширить данный протокол, должны (MUST) определить блок данных Идентификатор Поставщика, чтобы анонсировать способность реализовать расширения в черновом варианте проекта Internet. Предполагается, что черновым вариантам проектов Internet, которые получат одобрение и начнут стандартизоваться, будут присвоены "системные коды" из диапазона Future Use IANA и что требование использования идентификатора поставщика исчезнет.

Поля блока данных Идентификатор Поставщика определяются следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                        Vendor ID (VID)                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 18. Формат блока данных Идентификатор Поставщика.

Vendor ID - идентификатор поставщика (переменной длины) - Ответственность и гарантия уникальности, несмотря на отсутствие какого-либо центрального регистрирующего органа, лежит на том лице, которое выбирает идентификатор поставщика. Хорошей практикой считается включение названия компании, имени личности или чего-то подобного. Если вы хотите пустить пыль в глаза, вы можете включить широту, долготу и время того места, где вы были, когда выбирали идентификатор, а также некоторое случайное число. Дайджест сообщения от длинной уникальной строки более предпочтителен по сравнению с самой длинной уникальной строкой.

Тип блока данных Идентификатор Поставщика имеет значение сорок три (43).

3.13. Блок данных Селекторы Трафика

Блок данных Селекторы Трафика (Traffic Selector Payload), обозначаемый в данном меморандуме TS, позволяет партнерам идентифицировать потоки пакетов для обработки службами безопасности IPsec. Блок данных Селекторы Трафика состоит из общего заголовка блока данных IKE, за которым следуют отдельные селекторы трафика следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Number of TSs !                 RESERVED                      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       <Traffic Selectors>                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 19. Формат блока данных Селекторы Трафика.

- o Number of TSs - количество TS (1 октет) - Количество представленных селекторов трафика.

- o RESERVED - зарезервировано - Это поле должно (MUST) посылаться нулевым и должно (MUST) игнорироваться при приеме.

- o Traffic Selectors - селекторы трафика (переменной длины) - один или несколько отдельных селекторов трафика.

Длина блока данных Селекторы Трафика включает заголовок TS и все селекторы трафика.

Тип блока данных Селекторы Трафика имеет значение сорок четыре (44) для адресов на конце инициатора SA и сорок пять (45) для адресов на конце ответчика.

3.13.1. Селектор трафика

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   TS Type     !IP Protocol ID*|       Selector Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Start Port*         |           End Port*           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                         Starting Address*                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                         Ending Address*                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 20. Селектор трафика.

* Примечание: Все поля, за исключением полей TS Type и Selector Length, зависят от значения поля TS Type. Показанные поля соответствуют типам селекторов трафика 7 и 8, в настоящее время определено только два значения.

- o TS Type - тип селектора трафика (1 октет) - Определяет тип селектора трафика.

- o IP protocol ID - идентификатор протокола IP (1 октет) - Значение, определяющее связанный с ним идентификатор IP-протокола (например, UDP/TCP/ICMP). Нулевое значение означает, что идентификатор протокола не важен для данного селектора трафика - по SA могут передаваться все протоколы.

- o Selector Length - длина селектора - Определяет длину данной подструктуры селектора трафика (Traffic Selector Substructure), включая заголовок.

- o Start Port - начальный порт (2 октета) - Значение, определяющее наименьший номер порта, разрешаемый данным селектором трафика. Для протоколов, в которых номер порта не определяется, или в случае, если допускаются все порты, это поле должно (MUST) быть нулевым. Для протокола ICMP два однооктетных поля Type и Code (с полем Type в более значимых восьми битах и полем Code в менее значимых восьми битах) для целей фильтрации, основанной на этом поле, рассматриваются как один 16-битовый целый номер порта.

- End Port – оконечный порт (2 октета) - Значение, определяющее наибольший номер порта, разрешаемый данным селектором трафика. Для протоколов, в которых номер порта не определяется, или в случае, если допускаются все порты, это поле должно (MUST) быть равно 65535. Для протокола ICMP два однооктетных поля Type и Code (с полем Type в более значимых восьми битах и полем Code в менее значимых восьми битах) для целей фильтрации, основанной на этом поле, рассматриваются как один 16-битовый целый номер порта.

- Starting Address – начальный адрес – Наименьший адрес, включенный в данный селектор трафика (длина определяется типом селектора трафика).

- Ending Address – конечный адрес – Наибольший адрес, включенный в данный селектор трафика (длина определяется типом селектора трафика).

Системы, соответствующие [RFC4301] и желающие указать для портов признак "ANY" (любые), должны (MUST) устанавливать поле начального порта в ноль, а поле конечного порта в 65535; заметим, что в соответствии с [RFC4301], признак "ANY" включает признак "OPAQUE" (непрозрачный). Системы, работающие в соответствии с [RFC4301] и желающие указать порты типа "OPAQUE", а не порты типа "ANY", должны (MUST) устанавливать поле начального порта равным 65535, а поле конечного порта в 0.

В следующей таблице перечислены присвоенные значения для поля Traffic Selector Type и соответствующие данные адресных селекторов трафика (Address Selector Data).

Тип селектора трафика	Значение
RESERVED (Зарезервировано)	0-6
TS_IPV4_ADDR_RANGE (Диапазон адресов IPv4) Диапазон адресов IPv4, представленный двумя четырехоктетными значениями. Первое значение представляет собой начальный IPv4-адрес (включая его самого), а второе значение представляет собой конечный IPv4-адрес (включая его самого). Все адреса, попадающие между двумя определенными адресами, считаются находящимися в этом списке.	7
TS_IPV6_ADDR_RANGE (Диапазон адресов IPv6) Диапазон адресов IPv6, представленный двумя шестнадцатиоктетными (16) значениями. Первое значение представляет собой начальный IPv6-адрес (включая его самого), а второе значение представляет собой конечный IPv6-адрес (включая его самого). Все адреса, попадающие между двумя определенными адресами, считаются находящимися в этом списке.	8
RESERVED TO IANA (Зарезервировано для IANA)	9-240
PRIVATE USE (Частного пользования)	241-255

3.14. Блок данных Шифр

Блок данных Шифр (Encrypted Payload), обозначаемый в данном меморандуме SK{...}, содержит другие блоки данных в зашифрованном виде. Если блок данных Шифр появляется в сообщении, то он должен (MUST) быть последним блоком данных сообщения. Часто такой блок данных является единственным в сообщении.

Алгоритмы для шифрования и защиты целостности согласуются в процессе установки IKE_SA, а ключи вычисляются так, как определено в подразделах 2.14 и 2.18.

Алгоритмы для шифрования и защиты целостности создаются в соответствии с алгоритмами ESP, описанными в RFC 2104 [KBC96], 4303 [RFC4303] и 2451 [ESPCBC]. Данный документ полностью определяет криптографическую обработку данных IKE, но за логическим обоснованием следует обращаться к этим документам. Мы требуем использования блочного шифра с фиксированным размером блока и алгоритма проверки целостности, который вычисляет контрольную сумму фиксированной длины для сообщения переменной длины.

Тип блока данных Шифр имеет значение сорок шесть (46). Блок данных Шифр состоит из общего заголовка блока данных IKE, за которым следуют отдельные поля следующим образом:

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                     Initialization Vector                     !
!         (length is block size for encryption algorithm)       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                    Encrypted IKE Payloads                     !
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!               !             Padding (0-255 octets)            !
+-+-+-+-+-+-+-+-+                               +-+-+-+-+-+-+-+-+
!                                               !  Pad Length   !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                    Integrity Checksum Data                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 21. Формат блока данных Шифр.

- o Next Payload - следующий блок данных - Тип первого вложенного блока данных. Заметим, что это исключение для формата стандартного заголовка, поскольку блок данных Шифр является последним блоком данных в сообщении и, поэтому обычно это поле было бы равно нулю. Но поскольку содержимым данного блока данных являются вложенные блоки данных, и для размещения типа первого блока не было естественного места, этот тип размещается здесь.

- o Payload Length - длина блока данных - Включает длину заголовка, IV (вектора инициализации), зашифрованных блоков данных IKE, заполнителя, длины заполнителя и данных контрольной суммы целостности.

- o Initialization Vector - вектор инициализации - Случайно выбранное значение, длина которого равна длине блока основного алгоритма шифрования. Получатели должны (MUST) принимать любое значение. Отправители должны (SHOULD) либо выбирать это значение псевдослучайно и независимо для каждого сообщения, либо использовать последний блок зашифрованного текста последнего посланного сообщения. Отправители не должны (MUST NOT) использовать одно и то же значение для каждого сообщения, использовать последовательность значений с малым расстоянием Хемминга (например, порядковый номер) или использовать зашифрованный текст из полученного сообщения.

- o IKE Payloads - блоки данных IKE - представляют собой блоки данных, ранее специфицированные в данном разделе. Это поле зашифровывается согласованным шифром.

o Padding – заполнитель – может (MAY) содержать любое значение, выбранное отправителем, и должен (MUST) иметь длину, которая делает длину объединения блоков данных, заполнителя и длины заполнителя кратной размеру блока шифрования. Это поле зашифровывается согласованным шифром.

o Pad Length – длина заполнителя – представляет собой длину поля заполнителя. Отправитель должен (SHOULD) устанавливать длину заполнителя равной минимальному значению, которое делает длину объединения блоков данных, заполнителя и длины заполнителя кратной размеру блока, но получатель должен (MUST) принимать любую длину, которая обеспечивает надлежащее выравнивание. Это поле зашифровывается согласованным шифром.

o Integrity Checksum Data – данные контрольной суммы целостности – представляют собой криптографическую контрольную сумму всего сообщения, начиная с фиксированного заголовка IKE и заканчивая полем длины заполнителя. Должна (MUST) вычисляться контрольная сумма всего зашифрованного сообщения. Длина определяется согласованным алгоритмом целостности.

3.15. Блок данных Конфигурирование

Блок данных Конфигурирование (Configuration Payload), обозначаемый в данном документе CP, используется для обмена между партнерами IKE конфигурационной информацией. Этот обмен выполняется IRAC для запроса у IRAS внутреннего IP-адреса, а также для обмена другой информацией такого рода, которую можно получить с помощью протокола динамического конфигурирования хостов DHCP, как если бы IRAC был непосредственно подсоединен к ЛВС.

Блоки данных Конфигурирование имеют тип CFG_REQUEST/CFG_REPLY или CFG_SET/CFG_ACK (см. ниже CFG Type в описании блока данных). Блоки данных CFG_REQUEST и CFG_SET могут факультативно добавляться к любому запросу IKE. Ответ IKE должен (MUST) включать либо соответствующий блок данных CFG_REPLY или CFG_ACK, либо блок данных Уведомление (Notify) с типом ошибки, указывающим, почему запрос не мог быть принят. Исключением является то, что минимальная реализация может (MAY) игнорировать все блоки данных CFG_REQUEST и CFG_SET, так что ответное сообщение без соответствующих блоков данных CFG_REPLY или CFG_ACK должно (MUST) быть принято как указание того, что такой запрос не поддерживается.

Блоки данных "CFG_REQUEST/CFG_REPLY" позволяют оконечной точке IKE запросить информацию у своего партнера. Если атрибут в блоке данных Конфигурирование CFG_REQUEST имеет ненулевую длину, то он рассматривается как предложение для этого атрибута. Блок данных Конфигурирование CFG_REPLY может (MAY) вернуть либо это, либо новое значение. Он может (MAY) также содержать новые дополнительные атрибуты и не включать некоторые запрошенные атрибуты. Запросчики должны (MUST) игнорировать те возвращенные атрибуты, которые они не распознают.

Некоторые атрибуты могут (MAY) иметь несколько значений, в этом случае посылаются и/или возвращаются несколько значений атрибута одного и того же типа. Как правило, когда запрашивается атрибут, возвращаются все значения атрибута. Для некоторых атрибутов (в данной версии спецификации такими атрибутами являются только внутренние адреса) несколько запросов указывают на запрос атрибута, которому должны быть присвоены несколько значений. Для таких атрибутов количество возвращенных значений не должно (SHOULD NOT) превышать количества запрошенных.

Если тип данных, запрошенный в CFG_REQUEST, не распознается или не поддерживается, ответчик не должен (MUST NOT) возвращать тип ошибки, а вместо этого должен (MUST) либо послать CFG_REPLY, который может (MAY) быть пустым, либо ответ вообще не содержащий блока данных CFG_REPLY. Возвраты типов ошибок резервируются для тех случаев, когда запрос распознается, но не может быть выполнен в соответствии с требованиями, или когда запрос имеет неправильный формат.

Блоки данных "CFG_SET/CFG_ACK" позволяют оконечной точке IKE настоять на использовании конфигурационных данных ее партнером. В этом случае блок данных CFG_SET содержит атрибуты, которые партнер должен изменить по требованию инициатора. Ответчик должен (MUST) вернуть блок данных Конфигурирование, если он принял какие-либо конфигурационные данные, и этот блок должен (MUST) содержать атрибуты, которые принял ответчик с нулевой длиной данных. Те атрибуты, которые он не принял, не должны (MUST NOT) находиться в блоке данных Конфигурирование CFG_ACK. Если ни один из атрибутов не был принят, ответчик должен (MUST) вернуть либо пустой блок данных CFG_ACK, либо ответное сообщение без блока данных CFG_ACK. В настоящее время использование обмена CFG_SET/CFG_ACK не определено, хотя он может использоваться вместе с расширениями, базирующимися на идентификаторах поставщиков. Минимальная реализация данной спецификации может (MAY) игнорировать блоки данных CFG_SET.

Расширения на основе блока данных CP не должны (SHOULD NOT) использоваться для универсального управления. Их основное назначение заключается в предоставлении механизма начальной загрузки для обмена информацией от IRAS к IRAC в рамках IPsec. Хотя использование такого метода для обмена информацией между некоторыми защитными шлюзами (SGW - Security Gateway) или небольшими сетями может (MAY) оказаться полезным, для управления на уровне предприятия следует отдавать предпочтение существующим протоколам управления, например, DHCP [DHCP], RADIUS [RADIUS], SNMP или LDAP [LDAP], а также возникающим в результате их применения обменам информацией.

Блок данных Конфигурирование определяется следующим образом:

```
                     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  !C! RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   CFG Type    !                    RESERVED                   !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Configuration Attributes                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 22. Формат блока данных Конфигурирование.

Тип блока данных Конфигурирование имеет значение сорок семь (47).

- o CFG Type – тип конфигурирования (1 октет) – Тип обмена, представленный атрибутами конфигурирования.

```
CFG Type       Value
===========    =====
RESERVED         0
CFG_REQUEST      1
CFG_REPLY        2
CFG_SET          3
CFG_ACK          4
```

  Значения 5-127 зарезервированы для IANA. Значения 128-255 предназначены для частного пользования между взаимно согласными сторонами.

- o RESERVED – зарезервировано (3 октета) – Должны (MUST) посылаться нулевыми; должны (MUST) игнорироваться при приеме.

- o Configuration Attributes - атрибуты конфигурирования (переменной длины) – Имеются специфические для каждого блока данных Конфигурирование тип, длина и значение, которые определяются ниже. В блоке данных Конфигурирование может быть ноль или несколько атрибутов конфигурирования.

3.15.1. Атрибуты конфигурирования

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!R|         Attribute Type      !           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                             Value                             ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 23. Формат атрибута конфигурирования.

- o Reserved – зарезервировано (1 бит) – Этот бит должен (MUST) устанавливаться в ноль и должен (MUST) игнорироваться при приеме.

- o Attribute Type – тип атрибута - (15 бит) – Уникальный идентификатор для каждого типа атрибута конфигурирования.

- o Length – длина (2 октета) – Длина значения в октетах.

- o Value – значение (0 или больше октетов) – Значение переменной длины данного атрибута конфигурирования.

Определены следующие типы атрибутов конфигурирования:

Тип атрибута	Значение	Имеет несколько значений	Длина
RESERVED	0		
INTERNAL_IP4_ADDRESS	1	ДА*	0 или 4 октета
INTERNAL_IP4_NETMASK	2	НЕТ	0 или 4 октета
INTERNAL_IP4_DNS	3	ДА	0 или 4 октета
INTERNAL_IP4_NBNS	4	ДА	0 или 4 октета
INTERNAL_ADDRESS_EXPIRY	5	НЕТ	0 или 4 октета
INTERNAL_IP4_DHCP	6	ДА	0 или 4 октета
APPLICATION_VERSION	7	НЕТ	0 или более октетов
INTERNAL_IP6_ADDRESS	8	ДА*	0 или 17 октетов
RESERVED	9		
INTERNAL_IP6_DNS	10	ДА	0 или 16 октетов
INTERNAL_IP6_NBNS	11	ДА	0 или 16 октетов
INTERNAL_IP6_DHCP	12	ДА	0 или 16 октетов
INTERNAL_IP4_SUBNET	13	ДА	0 или 8 октетов
SUPPORTED_ATTRIBUTES	14	НЕТ	Кратна 2
INTERNAL_IP6_SUBNET	15	Да	17 октетов

* Эти атрибуты при возврате могут иметь несколько значений, только если было запрошено несколько значений.

Типы 16-16383 резервируются для IANA. Значения 16384-32767 предназначены для частного пользования между взаимно согласными сторонами.

- o INTERNAL_IP4_ADDRESS (внутренний IPv4-адрес), INTERNAL_IP6_ADDRESS (внутренний IPv6-адрес) – Адрес во внутренней сети, иногда называемый сокращенным адресом узла (red node address) или частным адресом (private address), и может (MAY) быть частным адресом в Internet. В сообщении запроса указанный адрес представляет собой запрашиваемый адрес (или ноль, если никакой конкретный адрес не запрашивается). Если запрашивается конкретный адрес, он вероятно указывает, что ранее существовало соединение с этим адресом, и что запросчик хотел бы повторно использовать этот адрес. В IPv6 запросчик может (MAY) представлять младшие байты адреса, который он хочет использовать. Путем запроса нескольких атрибутов типа внутренний адрес может (MAY) быть запрошено несколько внутренних адресов. Ответчик может (MAY) послать только не более запрошенного количества адресов. Внутренний IPv6-адрес (INTERNAL_IP6_ADDRESS) составляется из двух полей; первое поле представляет 16 октетов IPv6-адреса, а второе – один октет длины префикса, как определено в [ADDRIPV6].

Запрашиваемый адрес является годным до тех пор, пока не истечет срок его годности, определяемый атрибутом INTERNAL_ADDRESS EXPIRY, или между партнерами не останется контекстов безопасности IKE_SA.

- INTERNAL_IP4_NETMASK (внутренняя IPv4-маска) – Внутренняя сетевая маска. В сообщениях запроса и ответа допускается только одна сетевая маска (например, 255.255.255.0) и она должна (MUST) использоваться только с атрибутом INTERNAL_IP4_ADDRESS.

- INTERNAL_IP4_DNS (внутренний IPv4-DNS), INTERNAL_IP6_DNS (внутренний IPv6-DNS) – Определяет адрес DNS-сервера в сети. Может (MAY) запрашиваться несколько DNS-серверов. Ответчик может (MAY) ответить нулевым или большим количеством атрибутов этого типа.

- INTERNAL_IP4_NBNS (внутренний IPv4-NBNS), INTERNAL_IP6_NBNS (внутренний IPv6-NBNS) – Определяет адрес в сети NetBios Name Server (WINS). Может (MAY) запрашиваться несколько NBNS-серверов. Ответчик может (MAY) ответить нулевым или большим количеством атрибутов этого типа.

- INTERNAL_ADDRESS_EXPIRY – Определяет количество секунд, в течение которых хост может использовать внутренний IP-адрес. Хост должен (MUST) обновить IP-адрес до истечения этого времени. В ответе может (MAY) находиться только один такой атрибут.

- INTERNAL_IP4_DHCP, INTERNAL_IP6_DHCP – Сообщает хосту, чтобы он осуществлял посылку всех запросов внутреннего DHCP на адрес, содержащийся в атрибуте. Может (MAY) запрашиваться несколько DHCP-серверов. Ответчик может (MAY) ответить нулевым или большим количеством атрибутов DHCP-сервера.

- APPLICATION_VERSION – Информация о версии или приложении IPsec-хоста. Это строка печатаемых символов ASCII, которая не (NOT) завершается нулем.

- INTERNAL_IP4_SUBNET – Защищенные подсети, которые это пограничное устройство защищает. Этот атрибут составляется из двух полей; первое поле задает IP-адрес, а второе – сетевую маску. Может (MAY) запрашиваться несколько подсетей. Ответчик может (MAY) ответить нулевым или большим количеством атрибутов этого типа.

- SUPPORTED_ATTRIBUTES – Когда этот атрибут используется в запросе, он должен (MUST) иметь нулевую длину, и определяет запрос к ответчику, чтобы он ответил всеми атрибутами, которые он поддерживает. Ответ содержит атрибут, который содержит множество двухоктетных идентификаторов атрибутов. Длина, деленная на 2 (октета), определяет количество поддерживаемых атрибутов, содержащихся в ответе.

- INTERNAL_IP6_SUBNET - Защищенные подсети, которые это пограничное устройство защищает. Этот атрибут составляется из двух полей; первое поле задает 16-октетный IPv6-адрес, а второе – однооктетную длину префикса, как определено в [ADDRIPV6]. Может (MAY) запрашиваться несколько подсетей. Ответчик может (MAY) ответить нулевым или большим количеством атрибутов этого типа.

Заметим, что в данном документе не даются рекомендации, каким образом реализация в действительности узнает, какую информацию посылать в ответе. Т.е. мы не рекомендуем какого-либо конкретного метода определения сервером IRAS, какой DNS-сервер должен быть возвращен запрашивающему клиенту IRAC.

3.16. Блок данных Протокол расширяемой аутентификации (EAP)

Блок данных Протокол расширяемой аутентификации (Extensible Authentication Protocol), обозначаемый в данном меморандуме EAP, позволяет осуществлять аутентификацию контекстов безопасности IKE_SA с помощью протокола, определенного в RFC 3748 [EAP], и последующих расширениях этого протокола. Полное множество приемлемых для этого блока данных значений определяется в других местах, но в данный документ включена небольшая сводка из RFC 3748, чтобы общие случаи были представлены в одном документе.

```
                     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  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       EAP Message                             ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рис. 24. Формат блока данных EAP.

Тип блока данных EAP имеет значение сорок восемь (48).

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!     Code      ! Identifier    !           Length              !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!     Type      ! Type_Data...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Рис. 25. Формат сообщений EAP.

- Code – код (1 октет) указывает является ли данное сообщение запросом (Request (1)), ответом (Response (2)), благоприятным исходом (Success (3)) или неудачным исходом (Failure (4)).

- Identifier – идентификатор (1 октет) используется в PPP чтобы отличать повторно воспроизводимые сообщения от сообщений, посылаемых повторно. Поскольку в IKE EAP работает поверх надежного протокола, здесь данное поле не выполняет никаких функций. В сообщении ответа этот октет должен (MUST) устанавливаться так, чтобы его можно было сопоставить с идентификатором соответствующего запроса. В других сообщениях в это поле может (MAY) устанавливаться любое значение.

- Length – длина (два октета) представляет собой длину сообщения EAP, и должна (MUST) быть на четыре октета меньше значения поля Payload Length инкапсулирующего блока данных.

- Type – тип (1 октет) присутствует, только если поле Code указывает Request (1) или Response (2). Для других кодов длина сообщения EAP должна (MUST) быть равна четырем октетам, а поля Type и Type_Data не должны (MUST NOT) присутствовать. В сообщении запроса (Request (1)) поле Type указывает тип запрашиваемых данных. В сообщении ответа (Response (2)) поле Type должно (MUST) соответствовать либо типу Nak, либо типу запрошенных данных. В RFC 3748 определены следующие типы:

  1 Identity (идентификатор)
  2 Notification (уведомление)
  3 Nak (Response Only) – отрицательное подтверждение (только в ответе)
  4 MD5-Challenge (MD5-запрос)

5 One-Time Password (OTP) (одноразовый пароль)
6 Generic Token Card (типовая маркерная карта)

- Type_Data – данные типа (переменной длины) варьируются в зависимости от типа запроса и связанного с ним ответа. Для получения информации о методах EAP, см. [EAP].

Заметим, что поскольку IKE передает указание идентификатора инициатора в сообщении 3 протокола, ответчик не должен (SHOULD NOT) посылать запросы идентификатора EAP. Однако инициатор должен (SHOULD) отвечать на такие запросы, если он их принимает.

4. Требования к соответствию

Чтобы гарантировать интероперабельность всех реализаций IKEv2, в дополнение к требованиям, перечисленным в других местах, имеются требования типа "MUST support". Конечно, IKEv2 является протоколом безопасности, и одной из его главных функций является разрешение успешного установления контекстов безопасности только авторизованным сторонам. Таким образом, конкретная реализация может конфигурироваться с любыми ограничениями, касающимися алгоритмов и доверенных центров, что будет препятствовать всеобщей интероперабельности реализаций.

Протокол IKEv2 разработан таким образом, чтобы допускать возможность создания минимальных реализаций, которые могут взаимодействовать со всеми реализациями, соответствующими данной спецификации. Имеется целый ряд факультативных возможностей, которые легко могут игнорироваться конкретной реализацией, если она не поддерживает такую функцию. Эти функции включают:

Способность согласовывать контексты безопасности (SA) через NAT и туннелировать получающиеся ESP SA в UDP.

Способность запрашивать временный IP-адрес на удаленном конце туннеля (и отвечать на такой запрос).

Способность поддерживать различные типы традиционной аутентификации.

Способность поддерживать размер окна, больший, чем единица.

Способность устанавливать несколько ESP SA и/или AH SA в рамках одного IKE_SA.

Способность переустанавливать контексты безопасности.

Чтобы гарантировать интероперабельность, все реализации должны (MUST) быть способными разбирать все типы блоков данных (только для того, чтобы их пропустить) и игнорировать типы блоков данных, которые они не поддерживают, если только в заголовке блока данных не установлен бит critical. Если бит critical устанавливается в неподдерживаемом заголовке блока данных, все реализации должны (MUST) отбрасывать сообщения, содержащие такие блоки данных.

Каждая реализация должна (MUST) быть способной выполнять обмены IKE_SA_INIT и IKE_AUTH (состоящие из четырех сообщений), устанавливающие два контекста безопасности (один для IKE и один для ESP и/или AH). Реализации могут (MAY) быть только инициаторами или только ответчиками в соответствии со своими платформами. Каждая реализация должна (MUST) быть способной отвечать на информационный (INFORMATIONAL) обмен, но минимальная реализация может (MAY) отвечать на любое информационное (INFORMATIONAL) сообщение пустым информационным (INFORMATIONAL) ответом (заметим, что в контексте IKE_SA, "пустое" сообщение включает IKE-заголовок, за которым следует блок данных Шифр, не содержащий никаких блоков данных). Минимальная конфигурация может (MAY) поддерживать обмен CREATE_CHILD_SA только до такой степени, чтобы распознавать запросы и отклонять их блоком данных Уведомление (Notify) типа NO_ADDITIONAL_SAS. От минимальной реализации не требуется, чтобы она была способной инициировать обмен CREATE_CHILD_SA или информационный (INFORMATIONAL) обмен. Когда истекает SA, (созданный на основе локально конфигурируемых значений времени жизни или количества переданных октетов), реализация может (MAY) либо попытаться его обновить с помощью обмена CREATE_CHILD_SA, либо может (MAY) удалить (закрыть) старый SA и открыть новый SA. Если ответчик отвергает запрос CREATE_CHILD_SA с уведомлением NO_ADDITIONAL_SAS, то реализация должна (MUST) быть способной вместо этого закрыть старый SA и открыть новый SA.

От реализаций не требуется поддержка возможности запроса временных IP-адресов или ответа на такие запросы. Если реализация осуществляет поддержку, выдавая такие запросы, она должна (MUST) включать в сообщение 3 блок данных CP, содержащий по крайней мере поле типа INTERNAL_IP4_ADDRESS или INTERNAL_IP6_ADDRESS. Все другие поля являются факультативными. Если реализация поддерживает ответы на такие сообщения, она должна (MUST) осуществлять разбор блока данных CP типа CFG_REQUEST в сообщении 3 и распознавать поле типа INTERNAL_IP4_ADDRESS или INTERNAL_IP6_ADDRESS. Если она поддерживает аренду адреса соответствующего типа, она должна (MUST) возвращать блок данных CP типа CFG_REPLY, содержащий адрес запрошенного типа. Ответчик должен (SHOULD) включать в этот блок данных все другие связанные атрибуты, если он их имеет.

Минимальная реализация IPv4-ответчика будет игнорировать содержимое блока данных CP за исключением определения того, что он включает атрибут INTERNAL_IP4_ADDRESS, и будет отвечать адресом и другими связанными атрибутами независимо от того, запросил ли их инициатор.

Минимальная реализация IPv4-инициатора будет генерировать блок данных CP, содержащий только атрибут INTERNAL_IP4_ADDRESS, и будет разбирать ответ, игнорируя атрибуты, которые не знает, как использовать. Единственным атрибутом, который она должна (MUST) быть способной обрабатывать, является атрибут INTERNAL_ADDRESS_EXPIRY, который она должна использовать для ограничения времени жизни SA, если она не осуществит успешного обновления аренды до того, как истечет его время жизни. От минимальных инициаторов не требуется способность запрашивать обновление аренды, а минимальные ответчики могут не отвечать на такие запросы.

Чтобы реализацию можно было назвать соответствующей данной спецификации, должна (MUST) существовать возможность ее конфигурирования для приема следующего:

Сертификаты PKIX, содержащие и подписанные ключами RSA размером 1024 или 2048 бит, в которых в качестве идентификатора передается либо ID_KEY_ID, либо ID_FQDN, либо ID_RFC822_ADDR, либо ID_DER_ASN1_DN.

Аутентификация на основе общего ключа, в которой в качестве идентификатора передается либо ID_KEY_ID, либо ID_FQDN, либо ID_RFC822_ADDR.

Аутентификация, в которой ответчик аутентифицируется с помощью сертификатов PKIX, а инициатор – с помощью аутентификации на основе общего ключа.

5. Анализ безопасности

Хотя данный протокол разрабатывается так, чтобы минимизировать раскрытие конфигурационной информации не аутентифицированным партнерам, некоторого такого раскрытия избежать невозможно. Либо один, либо другой партнер должен сначала идентифицировать себя и доказать свою подлинность первым. Чтобы избежать зондирования, требуется, чтобы инициатор обмена идентифицировал себя первым, и обычно требуется, чтобы он аутентифицировал себя первым. Однако инициатор может узнать, что ответчик поддерживает IKE, а также какие криптографические алгоритмы он поддерживает. Ответчик (или кто-то, имитирующий ответчика) может зондировать инициатора не только на предмет его идентификатора, но с помощью блоков данных CERTREQ он может оказаться способным определить, какие сертификаты инициатор готов использовать.

Использование аутентификации на основе EAP некоторым образом меняет возможности зондирования. Когда используется аутентификация на основе EAP, ответчик доказывает свою подлинность до инициатора, таким образом инициатор, который знает имя действительного инициатора, может зондировать как имя, так и сертификаты ответчика.

Повторяющиеся переустановки ключей с помощью обмена CREATE_CHILD_SA без дополнительных обменов Диффи-Хеллмана оставляют все SA уязвимыми для криптоанализа одного ключа или для затопления любой из оконечных точек. Разработчики реализаций должны взять этот факт на заметку и установить ограничение на обмены CREATE_CHILD_SA между операциями возведения в степень. Данный меморандум не предписывает такого ограничения.

Стойкость ключа, полученного в результате обмена Диффи-Хеллмана с помощью любой определенной здесь группы, зависит от собственной стойкости группы, размера используемого показателя степени и энтропии, обеспечиваемой используемым генератором случайных чисел. Вследствие этих условий трудно определить стойкость ключа для любой определенной группы. Группа Диффи-Хеллмана номер два при использовании со стойким генератором случайных чисел и показателем степени размером не менее 200 бит является общепринятой для использования с 3DES. Группа пять обеспечивает большую безопасность, чем группа два. Группа один присутствует только по историческим причинам и не обеспечивает достаточной стойкости за исключением использования с DES, который тоже оставлен только по историческим причинам. Реализации должны взять на заметку эти оценки при установке политики и согласовании параметров безопасности.

Заметим, что эти ограничения распространяются только на сами группы Диффи-Хеллмана. В IKE отсутствует что-либо, что будет уменьшать стойкость, полученную от более стойких групп (ограниченную стойкостью других согласованных алгоритмов, включая функцию prf). В действительности, гибкая система IKE поддерживает определение большего числа групп; использование групп эллиптических кривых может значительно увеличить стойкость при использовании намного меньших чисел.

Предполагается, что все показатели степени Диффи-Хеллмана стираются из памяти после использования. В частности, эти показатели степени не должны (MUST NOT) получаться из долговременных секретов, подобных начальному значению псевдослучайного генератора, который не стирается после использования.

Стойкость всех ключей ограничена размером выходных данных согласованной функции prf. По этой причине, функция prf, выходные данные которой меньше 128 бит (например, 3DES-CBC) с данным протоколом не должна (MUST NOT) использоваться.

Безопасность данного протокола существенно зависит от случайности случайно выбираемых параметров. Они должны генерироваться строго случайным источником или псевдослучайным источником с должным образом подобранным начальным значением (см. [RFC4086]). Разработчики реализаций должны проявлять осторожность, чтобы гарантировать, что использование простых чисел как для ключей, так и для одноразовых номеров организуется таким образом, что не подрывает безопасности ключей.

Информацию по логическому обоснованию многих альтернатив криптографических свойств данного протокола см. в [SIGMA] и [SKEME]. Хотя безопасность согласованных CHILD_SA не зависит от стойкости алгоритмов шифрования и защиты целостности, согласованных в IKE_SA, реализации не должны (MUST NOT) согласовывать NONE в качестве алгоритма защиты целостности IKE, или ENCR_NULL в качестве алгоритма шифрования IKE.

При использовании предварительно распределенных ключей критическим соображением является то, как гарантировать случайность этих секретов. Наиболее сильная практика заключается в том, чтобы гарантировать, что любой заранее распределенный ключ содержал настолько много случайности, что и наиболее стойкий согласуемый ключ. Формирование общего секрета на основе пароля, имени или другого источника с низкой энтропией не безопасно. Эти источники являются объектом атак по словарю и атак социальной инженерии, а также ряда других атак.

Уведомления NAT_DETECTION_*_IP содержат значения хэш-функции от адресов и портов в попытке скрыть внутренние IP-адреса за NAT. Поскольку пространство IPv4-адресов представляется только 32 битами, и обычно оно очень разрежено, злоумышленник может выяснить внутренний адрес, используемый за устройством NAT, путем опробования всех возможных IP-адресов и попытки найти соответствующее значение хэш-функции. Номера портов обычно устанавливаются равными 500, а SPI могут быть извлечены из пакета. Это сокращает количество вычислений значений хэш-функций до 2^32. С известным предположением об использовании пространства частных адресов, количество вычислений значений хэш-функции намного меньше. Поэтому разработчики не должны предполагать, что использование IKE предотвратит утечку информации о внутренних адресах.

При использовании метода аутентификации EAP, который не генерирует общего ключа для защиты последующего блока данных AUTH, возможны определенного вида атаки типа "человек посередине" и имитации сервера [EAPMITM]. Эти уязвимости возникают, когда EAP используется также в протоколах, которые не защищаются безопасным туннелем. Поскольку EAP представляет собой универсальный протокол аутентификации, который часто используется для обеспечения средств односторонней подписи (single-signon), развернутое решение IPsec, которое основывается на методе аутентификации EAP, который не генерирует общего ключа (известный также под названием метод EAP, не генерирующий ключ), может оказаться скомпрометированным по причине развертывания полностью несвязанного приложения, которое также использует тот же самый не генерирующий ключей метод EAP, но в незащищенном режиме. Заметим, что эта уязвимость не ограничивается только EAP, но может возникать в других сценариях, в которых повторно используется инфраструктура аутентификации. Например, если механизм EAP, используемый IKEv2, употребляет маркерное удостоверение (token authenticator), злоумышленник – "человек посередине" – может имитировать web-сервер, перехватить обмен аутентификации маркерами и использовать его для инициирования соединения IKEv2. По этой причине, использование методов EAP, которые не генерируют ключей, необходимо (SHOULD) по возможности избегать. Там, где они используются, чрезвычайно важно, чтобы все применения этих методов EAP использовали (SHOULD) защищенный туннель, при котором инициатор проверяет сертификат ответчика до инициирования обмена EAP. Разработчики реализаций должны (SHOULD) описывать уязвимости использования не генерирующих ключей методов EAP в документации на свои реализации так чтобы администраторы, развертывающие решения IPsec, были осведомлены об этих опасностях.

Реализация, использующая EAP, должна (MUST) также использовать аутентификацию с открытым ключом сервера своему клиенту до того, как начнется обмен EAP, даже если метод EAP предлагает взаимную аутентификацию. Это позволяет избежать дополнительных вариаций протокола IKEv2 и защищает данные EAP от активных злоумышленников.

Если сообщения IKEv2 достаточно длинные, так что фрагментация на уровне IP является необходимой, возможно, что злоумышленники могут не дать завершить обмен путем исчерпания буферов повторной сборки. Возможность такой атаки можно минимизировать путем использования кодирования типа Hash and URL вместо посылки сертификатов (см. подраздел 3.6). Дополнительные упрощения обсуждаются в [KPS03].

6. Соображения для IANA

В данном документе определяется целый ряд новых типов полей и значений, для которых будущие присваивания будут выполняться IANA.

Должны быть созданы следующие реестры:

- Типы обменов IKEv2 (подраздел 3.1)
- Типы блоков данных IKEv2 (подраздел 3.2)
- Типы преобразований IKEv2 (пункт 3.3.2)
  - Типы атрибутов преобразований IKEv2 (пункт 3.3.2)
  - Идентификаторы преобразований шифрования IKEv2 (пункт 3.3.2)
  - Идентификаторы преобразований псевдослучайных функций IKEv2 (пункт 3.3.2)

Идентификаторы преобразований алгоритмов целостности IKEv2 (пункт 3.3.2)
Идентификаторы преобразований Диффи-Хеллмана IKEv2 (пункт 3.3.2)
Типы идентификаторов блока данных идентификации IKEv2 (подраздел 3.5)
Кодировки сертификатов IKEv2 (подраздел 3.6)
Метод аутентификации IKEv2 (подраздел 3.8)
Типы сообщений Уведомление IKEv2 (пункт 3.10.1)
Идентификаторы уведомлений преобразований IPCOMP IKEv2 (пункт 3.10.1)
Идентификаторы протоколов безопасности IKEv2 (пункт 3.3.1)
Типы селекторов трафика IKEv2 (пункт 3.13.1)
Типы конфигурирования блоков данных конфигурирования IKEv2 (подраздел 3.15)
Типы атрибутов блоков данных конфигурирования IKEv2 (пункт 3.15.1)

Примечание: при создании нового типа преобразования для него должен создаваться новый реестр.

Изменения и дополнения к любому из этих реестров должны выполняться под наблюдением экспертов.

7. Благодарности

Данный документ является результатом совместных усилий всей рабочей группы IPsec. Если бы не было ограничений на количество авторов, которые могут заявляться в документах RFC, то были бы перечислены в алфавитном порядке следующие имена: Bill Aiello, Stephane Beaulieu, Steve Bellovin, Sara Bitan, Matt Blaze, Ran Canetti, Darren Dukes, Dan Harkins, Paul Hoffman, John Ioannidis, Charlie Kaufman, Steve Kent, Angelos Keromytis, Tero Kivinen, Hugo Krawczyk, Andrew Krywaniuk, Radia Perlman, Omer Reingold и Michael Richardson. В этот проект внесли свой вклад многие другие люди. Hugh Daniel предложил возможность указания инициатором в сообщении 3 имени ответчика и дал этой возможности изящное название "You Tarzan, Me Jane". David Faucher и Valery Smyzlov помогли усовершенствовать согласование селекторов трафика.

8. Ссылки

8.1. Нормативные ссылки

[ADDGROUP] Kivinen, T., and Kojo, M., "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[ADDRIPV6] Hinden, R., and Deering, S., "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

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

[EAP] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and Levkowetz, H., "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[ESPCBC] Pereira, R., Adams, R., "The ESP CBC-Mode Cipher Algorithms", RFC 2451, November 1998.

[Hutt05] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC3168] Ramakrishnan, K., Floyd, S., and Black, D., "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3280] Housley, R., Polk, W., Ford, W., Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

8.2. Информативные ссылки

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

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

[DHCP] R. Droms, "Dynamic Host Configuration Protocol", RFC2131

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

[EAPMITM] Asokan, N., Nierni, V., and Nyberg, K., "Man-in-the-Middle in Tunneled Authentication Protocols", http://eprint.iacr.org/2002/163, November 2002.

[HC98] Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

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

[IPCOMP] Shacham, A., Monsour, R., Pereira, R., and Thomas, M., "IP Payload Compression Protocol (IPComp)", RFC 3173, September 2001.

[KPS03] Kaufman, C., Perlman, R., and Sommerfeld, B., "DoS protection for UDP-based protocols", ACM Conference on Computer and Communications Security, October 2003.

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

[LDAP] M. Wahl, T. Howes, S. Kille., "Lightweight Directory Access Protocol (v3)", RFC 2251

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

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

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

[PFKEY] McDonald, D., Metz, C., and Phan, B., "PFKEY Key Management API, Version 2", RFC 2367, July 1998.

[PKCS1] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[PK01] Perlman, R., and Kaufman, C., "Analysis of the IPsec key exchange Standard", WET-ICE Security Conference, MIT,2001, http://sec.femto.org/wetice-2001/papers/radia-paper.pdf.

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

[RADIUS] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC4086] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC1958] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC2401] Kent, S., and Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F. and Black, D., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and Weiss, W., "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2522] Karn, P., and Simpson, W., "Photuris: Session-Key Management Protocol", RFC 2522, March 1999.

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, February 2000.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.

[RFC3439] Bush, R. and D. Meyer, "Some Internet Architectural Guidelines and Philosophy", RFC 3429, December 2002.

[RFC3715] Aboba, B and Dixon, W., "IPsec-Network Address Translation (NAT) Compatibility Requirements", RFC 3715, March 2004.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

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

[SHA] NIST, "Secure Hash Standard", FIPS 180-1, National

Institute of Standards and Technology, U.S. Department of Commerce, May 1994.

[SIGMA] Krawczyk, H., "SIGMA: the `SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols", in Advances in Cryptography - CRYPTO 2003 Proceedings, LNCS 2729, Springer, 2003. Available at: http://www.informatik.uni-trier.de/~ley/db/conf/crypto/crypto2003.html.

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

[X.501] ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models, 1993.

[X.509] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.

Приложение А. Сводка изменений по сравнению с IKEv1

Целями данной ревизии IKE являются:

1) Определить весь протокол IKE в одном документе, который заменит RFC 2407, 2408 и 2409, и включит в него более поздние изменения для поддержки функций пересечения NAT, расширяемой аутентификации и удаленного получения адреса.

2) Упростить IKE путем замены восьми различных начальных обменов одним обменом из четырех сообщений (с изменениями в механизмах аутентификации, влияющими только на один блок данных AUTH, вместо изменения структуры всего обмена).

3) Ликвидировать поля Domain of Interpretation (DOI), Situation (SIT) и Labeled Domain Identifier, а также биты Commit и Authentication only;

4) Уменьшить задержку IKE в общем случае путем реализации начального обмена за два прохода туда и обратно (4 сообщения) и разрешения возможности комбинирования установки контекста безопасности CHILD_SA с этим обменом;

5) Для упрощения реализации и анализа безопасности заменить криптографический синтаксис для защиты самих сообщений IKE синтаксисом, тесно связанным с ESP,;

6) Сократить количество возможных ошибочных состояний путем реализации надежности (все сообщения подтверждаются) и упорядочивания протокола. Это позволяет укоротить обмены CREATE_CHILD_SA с трех до двух сообщений;

7) Повысить надежность путем разрешения ответчику не выполнять существенной обработки до тех пор, пока он не получит сообщения, доказывающего, что инициатор может получать сообщения на заявленный им IP-адрес, и не фиксировать какого-либо состояния обмена до тех пор, пока инициатор не сможет быть криптографически аутентифицирован;

8) Исправить криптографические слабости, такие как проблема с симметричностью в вычислении значений хэш-функций, используемых для аутентификации, которые были задокументированы Теро Кивиненом (Tero Kivenen);

9) Специфицировать селекторы трафика в их собственном типе блока данных, а не перегружать блоки данных идентификации, и сделать селекторы трафика более гибкими для специфицирования;

10) Специфицировать требуемое поведение при определенных ошибочных ситуациях или когда принимаются данные, которые не понимаются, чтобы упростить будущие ревизии таким способом, который не нарушает обратной совместимости;

11) Упростить и прояснить, как поддерживается общее состояние при наличии отказов в сети и атак типа отказа в обслуживании; и

12) По мере возможности поддерживать существующий синтаксис и магические числа, чтобы сделать его пригодным для возможного улучшения реализаций IKEv1 и для поддержки IKEv2 с минимальными усилиями.

Приложение В. Группы Диффи-Хеллмана

Здесь определяются две группы Диффи-Хеллмана для использования в IKE. Эти группы были сгенерированы Ричардом Шройппелем (Richard Schroeppel) из аризонского университета. Свойства этих простых чисел описываются в [Orm96].

Стойкости, предоставляемой группой 1, может быть не достаточно для обязательных для реализации алгоритмов шифрования, и она упоминается здесь по историческим причинам.

Дополнительные группы Диффи-Хеллмана определяются в [ADDGROUP].

B.1. Группа 1 – 768-битовый MODP

Этой группе присвоен идентификатор 1 (единица).

Простое число равно: 2^768 - 2 ^704 - 1 + 2^64 * { [2^638 pi] + 149686 }

Его шестнадцатеричное значение равно:

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

Генератор равен 2.

B.2. Группа 2 – 1024-битовый MODP

Этой группе присвоен идентификатор 2 (два).

Простое число равно 2^1024 - 2^960 - 1 + 2^64 * { [2^894 pi] + 129093 }.

Его шестнадцатеричное значение равно:

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

Генератор равен 2.

Адрес редактора

Charlie Kaufman
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052

Phone: 1-425-707-3335
EMail: charliek@microsoft.com

Полное определение авторских прав

Определение интеллектуальной собственности

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.