Network Working Group
Request for Comments: 4306
Obsoletes: 2407, 2408, 2409
Category: Standards Track

C. Kaufman, Ed.
Microsoft

December 2005

Протокол обмена ключами в Internet (IKEv2)

Internet Key Exchange (IKEv2) Protocol

Статус документа

В этом документе содержится спецификация протокола, предложенного сообществу Internet. Документ служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола вы можете узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться без ограничений.

Авторские права

Аннотация

В этом документе описана версия 2 протокола обмена ключами в Internet (IKE[1]). Протокол IKE является частью IPsec и служит для обоюдной аутентификации партнёров, организации и поддержки защищённых связей (SA[2]).

Эта версия спецификации IKE объединяет содержимое нескольких отдельных документов прежних версий, включая ISAKMP[3] (RFC 2408), IKE (RFC 2409), DOI[4] (RFC 2407), спецификация работы через NAT[5], унаследованную аутентификацию и получение удалённого адреса.

Версия 2 протокола IKE не совместима с версией 1, но имеет достаточно много общего в формате заголовков и обе версии однозначно могут работать через один и тот же порт UDP.

Оглавление

[1]Internet Key Exchange.
[2]Security association.
[3]Internet Security Association and Key Management Protocol - протокол управления ключами и защищёнными связями Internet.
[4]Domain of Interpretation - область интерпретации.
[5]Network Address Translation - трансляция сетевых адресов.

1. Введение

Протоколы защиты IP (IPsec[1]) обеспечивает конфиденциальность, целостность данных, контроль доступа, а также аутентификацию источника данных для дейтаграмм IP. Эти услуги обеспечиваются за счёт поддержки разделяемого состояния между источником и приёмником дейтаграмм IP. Это состояние, наряду с другими аспектами, определяет услуги, предоставляемые дейтаграммам, используемые для этих услуг криптографические алгоритмы, а также ключи, используемые на входе криптоалгоритмов.

Организация этого состояния вручную не обеспечивает приемлемого масштабирования. Следовательно, требуется протокол для динамической организации этого состояния. В данном документе описан такой протокол - протокол обмена ключами Internet (IKE). Данное описание относится к версии 2 протокола IKE. Первая версия протокола IKE была определена в RFC 2407, 2408 и 2409 [Pip98, MSST98, HC98]. Данный документ заменяет три упомянутых RFC.

Определения основополагающих терминов, используемых в документе (таких, как защищённая связь или SA) можно найти в [RFC4301].

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с [Bra97].

Термин Expert Review (экспертиза) интерпретируется в соответствии с определением [RFC2434].

IKE выполняет взаимную аутентификацию партнёров и организует защищённую связь IKE SA, включающую разделяемый секретный ключ, который может эффективно использоваться при организации SA для протоколов ESP[2] [RFC4303] и/или AH[3] [RFC4302], и набор криптографических алгоритмов, которые будут использоваться SA для защиты передаваемого трафика. В этом документе термины «набор» (suite) или «шифронабор» (cryptographic suite) обозначают все множество алгоритмов, используемых для защиты SA. Инициатор предлагает один или несколько наборов, перечисляя поддерживаемые им алгоритмы, которые могут быть объединены в наборы или использоваться «вперемешку». IKE может также согласовывать использование компрессии IP (IPComp[4]) [IPCOMP] совместно в ESP и/или AH SA. Для IKE SA будем использовать обозначение IKE_SA. SA для ESP и/или AH, проходящие через IKE_SA, будем обозначать CHILD_SA.

Весь обмен информацией IKE организован в форме парных сообщений запрос - отклик. Для пар используется термин «обмен» (exchange). Первые сообщения при организации IKE_SA включают обмен IKE_SA_INIT и IKE_AUTH, а за ними следуют обмены CREATE_CHILD_SA или INFORMATIONAL. В общем случае для организации IKE_SA и первой связи CHILD_SA используется один обмен IKE_SA_INIT и один обмен IKE_AUTH (всего 4 сообщения). В исключительных случаях оба обмена могут использоваться неоднократно. В любом случае все обмены IKE_SA_INIT **должны** быть

[1]IP Security.
[2]Encapsulating Security Payload - инкапсуляция защищённых данных.
[3]Authentication Header - аутентификационный заголовок.
[4]IP Compression.

завершены до начала обмена любого другого типа, после этого **должны** быть завершены все обмены IKE_AUTH и далее могут выполняться в любом порядке обмены CREATE_CHILD_SA и INFORMATIONAL. В некоторых сценариях между конечными точками IPsec требуется только один обмен CHILD_SA и, следовательно, не возникает дополнительных обменов. Последующие обмены **могут** использоваться для организации дополнительных CHILD_SA между теми же аутентифицированными парами конечных точек и для выполнения вспомогательных функций.

Поток сообщений IKE всегда состоит из запросов, за которыми следует соответствующий отклик. Ответственность за обеспечение надёжности возлагается на запрашивающую сторону. Если отклик не получен в течение заданного времени ожидания, запрашивающая сторона должна повторить запрос (или прервать попытку соединения).

Первый запрос/отклик в сеансе IKE (IKE_SA_INIT) согласует параметры защиты для IKE_SA, передаёт специальные сигналы nonce и значения Diffie-Hellman.

Вторая пара запрос/отклик (IKE_AUTH) передаёт аутентификацию, обеспечивает информацию о секретах аутентифицированных сторон и организует защищённую связь для первой (зачастую, единственной) AH или ESP CHILD_SA.

Последующие обмены относятся к типу CREATE_CHILD_SA (создание CHILD_SA) и INFORMATIONAL (удаление SA, сообщения об ошибках и другие служебные функции). Каждый запрос требует отклика. Запросы типа INFORMATIONAL, не содержащие информации (кроме пустого поля Encrypted payload, требуемого синтаксисом) обычно используются для проверки сохранности соединения. Последующие обмены не могут осуществляться, пока не будут завершены начальные обмены. Далее в описании предполагается отсутствие ошибок. Изменения потока, связанные с ошибками, рассмотрены в параграфе 2.21.

1.1. Сценарии использования

Предполагается, что IKE будет использоваться при согласовании SA для протоколов ESP и/или AH SA во множестве различных сценариев с отличающимися требованиями.

1.1.1. Туннель между защитными шлюзами

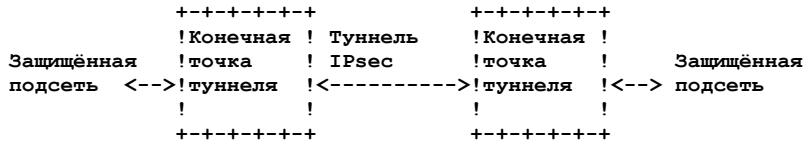

Рисунок 1. Туннель между защитными шлюзами.

В этом сценарии ни одна из конечных точек соединений IP не поддерживает IPsec, но расположенные между конечными точками узлы сетей обеспечивают защиту трафика на пути передачи. Защита прозрачна для конечных точек и пакеты через обычную систему маршрутизации передаются в конечную точку туннеля для обработки. Каждая из конечных точек туннеля будет анонсировать множество расположенных за ней адресов и пакеты для этих адресов будут передаваться в туннель с адресом конечного получателя во внутреннем заголовке IP.

1.1.2. Туннель между конечными точками

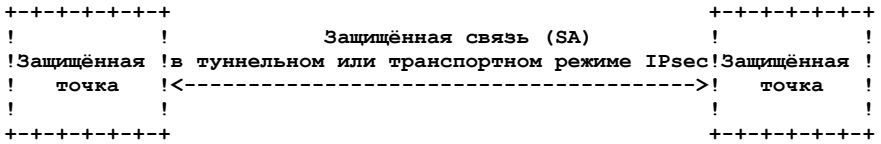

Рисунок 2. Туннель между конечными точкам.

В этом сценарии обе конечные точки соединения IP реализуют IPsec в соответствии с требованиями для хостов в [RFC4301]. Обычно используется транспортный режим без внутренних заголовков IP. Если внутренний заголовок используется, адрес IP в нем будет совпадать с адресом во внешнем заголовке. Для защиты с помощью данной SA согласуется одна пара адресов. Конечные точки **могут** реализовать средства контроля доступа на прикладных уровнях на основе IPsec-аутентификации участников соединения. Этот сценарий обеспечивает сквозную защиту, которая является одним из принципов работы Internet с момента разработки [RFC1958], [RFC2775] и метода ограничения унаследованных проблем, связанных со сложностью сетей, которые отмечены в [RFC3439]. Хотя этот сценарий не может полноценно применяться в Internet на базе IPv4, он может успешно использоваться внутри сетей intranet на базе IKEv1. Более широкому распространению этого сценария будет способствовать переход на IPv6 и адаптация IKEv2.

В таком сценарии одна или обе конечных точки могут находиться за системой трансляции сетевых адресов (NAT). В этом этом случае туннелируемые пакеты будут инкапсулироваться в UDP так, что номера портов в заголовках UDP можно будет использовать для аутентификации отдельных конечных точек, расположенных за NAT (см. параграф 2.23).

1.1.3. Туннель между конечной точкой и защитным шлюзом

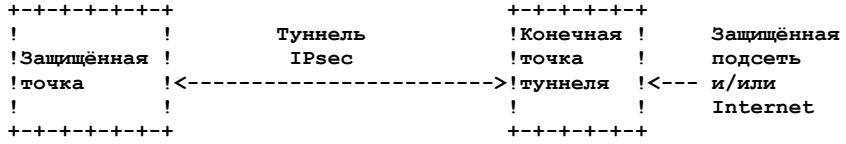

Рисунок 3. Туннель между конечной точкой и защитным шлюзом.

В этом сценарии защищённая конечная точка (обычно портативный компьютер) подключается к корпоративной сети с использованием защищённого туннеля IPsec. Туннель может использоваться только для доступа к информации, хранящейся в корпоративной сети, или служить для передачи всего трафика портативного компьютера через офисную сеть для обеспечения возможности использования корпоративного межсетевого экрана, защищающего компьютер от

атак из сети Internet. В любом случае защищённой точке будет нужен IP-связанный с защитным шлюзом, чтобы адресованные этой точке пакеты попадали на защитный шлюз и передавались им через туннель на защищённую точку. Этот адрес может выделяться защитным шлюзом статически или динамически. Во втором случае IKEv2 включает механизм, с помощью которого инициатор соединения запрашивает адрес IP, принадлежащий шлюзу, для использования в течение срока действия SA.

В этом сценарии пакеты будут использовать туннельный режим. В каждом пакете от защищённой конечной точки внешний заголовок IP будет содержать адрес отправителя, связанный с текущим местоположением (адрес, по которому трафик будет маршрутизироваться непосредственно к конечной точке), а внутренний заголовок IP будет содержать адрес отправителя, выделенный защитным шлюзом (т. е., адрес, по которому трафик будет маршрутизироваться защитному шлюзу для пересылке конечной точке). Внешний адрес получателя будет указывать защитный шлюз, а внутренний адрес получателя - конечного адресата пакета.

В этом сценарии защищённая конечная точка может находиться за системой трансляции адресов (NAT). При этом адрес, который будет видеть защитный шлюз, отличается от IP-адреса защищённой конечной точки и пакеты будут инкапсулироваться в дейтаграммы UDP для обеспечения корректной маршрутизации.

1.1.4. Другие сценарии

Обозначение	*Данные*
AUTH	аутентификация
CERT	сертификат
CERTREQ	запрос сертификата
CP	конфигурация
DD	удаление
E	зашифровано
EAP	расширенная аутентификация
HDR	заголовок IKE
IDi	идентификация - инициатор
IDr	идентификация - отвечающая сторона
KE	обмен ключами
Ni,Nr	Nonce
NN	уведомление
SA	защищённая связь
TSi	селектор трафика - инициатор
TSr	селектор трафика - отвечающая сторона
VV	идентификатор производителя

Возможны и другие сценарии, представляющие собой комбинации перечисленных выше вариантов. Один примечательный вариант объединяет в сете аспекты 1.1.1 и 1.1.3. Подсеть может организовать весь доступ наружу через удалённый защитный шлюз, используя туннель IPsec и тогда внешние сети (Internet) будут маршрутизировать пакеты для подсети защитному шлюзу. Например, некая домашняя сеть может виртуально представляться в сети Internet статическими адресами IP, несмотря на то, что эта сеть подключена через ISP, который выделяет один динамический адрес пользовательскому защитному шлюзу (видимый в Internet статический адрес IP и трансляция IPsec обеспечивается третьей стороной, расположенной в другом месте).

1.2. Начальные обмены

Коммуникации с использованием IKE всегда начинаются с обменов IKE_SA_INIT и IKE_AUTH (в IKEv1 - Фаза 1). Эти начальные обмены включают четыре сообщения, хотя в некоторых сценариях это число может расти. Все коммуникации с использованием IKE состоят из пар «запрос-отклик». Сначала будет описываться базовый обмен, а затем - возможные варианты. Первая пара сообщений (IKE_SA_INIT) согласует криптографические алгоритмы, осуществляет обмен сигналами nonce и обмен Diffie-Hellman [DH].

Вторая пара сообщений (IKE_AUTH) аутентифицирует предыдущие сообщения, обеспечивает обмен аутентификационной информацией и сертификатами, а также создаёт первую CHILD_SA. Компоненты этих сообщений шифруются и целостность их защищается с использованием ключей, организованных при обмене IKE_SA_INIT, поэтому аутентификационные данные недоступны для подслушивания, а все поля сообщений аутентифицируются.

На врезке справа приведён список обозначений и краткое описание данных, содержащихся в сообщениях.

Содержимое данных в сообщениях подробно рассматривается в разделе 3. Данные, которые являются необязательными, указываются в квадратных скобках - [CERTREQ] показывает возможность включения запроса сертификата.

Начальные обмены имеют вид:

```
      Инициатор                          Ответчик
     -----------                        ----------

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

HDR содержит списки параметров защиты (SPI[1]), номера версий и различные флаги. SAi1 указывает поддерживаемые инициатором криптографические алгоритмы для IKE_SA. В KE передаются значения Diffie-Hellman от инициатора. Ni задаёт nonce от инициатора.

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

Отвечающая сторона выбирает криптографический набор из числа предложенных инициатором и указывает свой выбор в SAr1, завершает обмен Diffie-Hellman в KEr и передаёт свой сигнал nonce в Nr.

На этом этапе согласования каждая из сторон может генерировать «затравку» SKEYSEED, на основе которой будут создаваться все ключи для данной IKE_SA. Все, кроме заголовков, во всех последующих сообщениях будет шифроваться с дополнительной защитой целостности. Ключи, используемые для шифрования и защиты целостности, создаются на основе SKEYSEED и обозначаются SK_e (encryption - шифрование) и SK_a (authentication -

[1] Security Parameter Index.

аутентификация для защиты целостности). Для каждого направления создаются отдельные ключи SK_e и SK_a. В дополнение к ключам SK_e и SK_a, создаваемым из значения DH для защиты IKE_SA, создаётся другая величина SK_d, которая используется для создания последующего материала для защищённых связей CHILD_SA. Обозначения SK { ... } показывают, что эти данные зашифрованы с защитой целостности на основе ключей SK_e и SK_a для соответствующего направления.

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

Инициатор предъявляет свою идентификацию в IDi, доказывает своё знание ключа, соответствующего IDi и защищает целостность содержимого первого сообщения, используя AUTH (см. параграф 2.15). Он может также передать свой сертификат (сертификаты) в CERT и список своих доверенных привязок[1] в CERTREQ. При включении CERT первый представляемый сертификат **должен** содержать открытый ключ, используемый для проверки поля AUTH. Необязательные данные IDr позволяют инициатору указать, какую идентификацию он хочет получить от отвечающей стороны. Это полезно в тех случаях, когда на машине, где работает отвечающая сторона, поддерживается множество вариантов идентификации для одного адреса IP. Инициатор начинает согласовывать CHILD_SA с использованием SAi2. Завершающие поля (начиная с SAi2) описаны при рассмотрении обмена CREATE_CHILD_SA.

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

Отвечающая сторона представляет свою идентификацию в IDr, и может передавать один или множество сертификатов (как и у инициатора, сертификат, содержащий публичный ключ для проверки AUTH, должен указываться первым), подтверждает свою идентификацию, защищает целостность второго сообщения с помощью AUTH и завершает согласование CHILD_SA в дополнительных полях, описанных ниже для обмена CREATE_CHILD_SA.

Получателя сообщений 3 и 4 **должны** проверить корректность расчёта всех сигнатур и MAC, а также соответствие имён в ID ключам, используемым для генерации AUTH.

1.3. Обмен CREATE_CHILD_SA

Этот обмен включает одну пару «запрос-отклик» и обозначается, как фаза 2 обмена в IKEv1. Обмен **может** инициироваться любой из сторон IKE_SA после завершения начальных обменов.

Все сообщения после первоначального обмена шифруются с использованием криптографического алгоритма и ключей, согласованных в первых двух сообщениях обмена IKE. Последующие сообщения используют синтаксис Encrypted Payload (зашифрованные данные), описанный в параграфе 3.14. Все последующие сообщения включаются в Encrypted Payload, даже если они указаны в тексте документа, как «пустые».

Любая из конечных точек может инициировать обмен CREATE_CHILD_SA, поэтому в данной секции термин «инициатор» означает конечную точку, начинающую этот обмен.

CHILD_SA создаётся путём передачи запроса CREATE_CHILD_SA. Этот запрос **может** содержать данные KE для дополнительного обмена Diffie-Hellman, позволяющего заблаговременно гарантировать более строгую защиту (секретность) для CHILD_SA. Ключевой материал для CHILD_SA является функцией SK_d, организованного при создании IKE_SA, обмена сигналами nonce в процессе обмена CREATE_CHILD_SA, и значения Diffie-Hellman (если данные KE включены в обмен CREATE_CHILD_SA).

В CHILD_SA, создаваемой, как часть начального обмена, **недопустимо** передавать второй KE и nonce. Сигналы nonce из начального обмена используются при расчёте ключей для CHILD_SA.

Запрос CREATE_CHILD_SA включает:

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

Инициатор передаёт предложение (предложения) SA в данных SA, nonce в Ni, может передать значение Diffie-Hellman в KEi, а также предложенные селекторы трафика в TSi и TSr. Если этот обмен CREATE_CHILD_SA служит для замены ключей существующей SA, отличной от IKE_SA, ведущие данные N типа REKEY_SA MUST аутентифицируют SA, для которой меняются ключи. Если этот обмен CREATE_CHILD_SA не является заменой ключей для существующей SA, данные N **должны** быть опущены. Если предложения SA включают различные группы Diffie-Hellman, данные KEi **должны** быть элементом группы, которую инициатор желает принять от отвечающей стороны. Если это предположение ошибочно, обмен CREATE_CHILD_SA завершится неудачей и будет повторяться с другим KEi.

Сообщение, следующее за заголовком, шифруется а для сообщения и заголовка обеспечивается защита целостности с использованием алгоритмов, согласованных для IKE_SA.

Отклик CREATE_CHILD_SA содержит:

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

Отвечающая сторона передаёт (используя то же значение Message ID) воспринятое предложение в данных SA, значение Diffie-Hellman в Ker, если в запрос были включены данные Kei, и выбранный криптографический набор, включающий данную группу. Если отвечающий выбирает криптографический набор с другой группой, он **должен** отвергнуть запрос. Инициатору **следует** повторить запрос с данными Kei из группы, выбранной отвечающим.

Селектор трафика для трафика, который будет передаваться в данной SA, указывается в данных TS, которые могут быть подмножеством предложенного инициатором CHILD_SA. Селекторы трафика опускаются, если данный запрос CREATE_CHILD_SA будет использоваться для смены ключа IKE_SA.

1.4. Обмен INFORMATIONAL

В различные моменты работы IKE_SA партнёры могут пожелать передать другой стороне управляющие сообщения, связанные с ошибками, или уведомления о некоторых событиях. Для обеспечения такой возможности в IKE определён

[1]Trust anchor.

информационный (INFORMATIONAL). Обмен INFORMATIONAL **должен** осуществляться **только** завершения начальных обменов и организации криптографической защиты с использованием согласованных ключей.

Управляющие сообщения, которые относятся к IKE_SA MUST, **должны** передаваться в данной IKE_SA. Управляющие сообщения, которые относятся к CHILD_SA **должны** передаваться под защитой IKE_SA, которая сгенерировала их (или её наследник, если IKE_SA была заменена при смене ключей).

Сообщения информационного обмена содержат 0 или более элементов данных Notification, Delete и Configuration. Получатель запроса INFORMATIONAL **должен** передать некий отклик (иначе отправитель будет предполагать потерю сообщения в сети и повторять его). Отклик **может** быть сообщением без элементов данных. Запросное сообщение в информационном обмене также **может** не содержать элементов данных. Предполагается, что это может использоваться конечными точками для проверки того, что партнёр «жив».

Защищённые связи ESP и AH всегда существуют в паре по одной SA для каждого направления. При закрытии SA **должны** быть закрыты оба члена пары. К тех случаях, когда SA являются вложенными, а также когда данные (и заголовки IP в туннельном режиме) сначала инкапсулируются с использованием IPComp, затем организовано ESP и, наконец, AH между одной парой конечных точек, все SA **должны** удаляться вместе. Каждая из конечных точек **должна** закрыть свои входящие SA и позволить другой точке закрыть соответствующую SA в каждой паре. Для удаления SA используется информационный обмен с передачей одного или множества элементов удаления (Delete Payload), перечисляющих SPI (которые ожидаются в заголовках входящих пакетов) удаляемых SA. Получатель **должен** закрыть означенные SA. Обычно отклик в информационном обмене будет содержать элементы удаления для парных SA обратного направления. Но существует одно исключение. Если обе стороны набора SA независимо решат закрыть их, каждая может передать элемент удаления и два запроса могут пересечься в сети. Если узел получает запрос удаления для SA, которые он уже указал в запросе удаления, он **должен** удалить исходящие SA в процессе обработки запроса и входящие SA при обработке отклика. В таких случаях в отклик **недопустимо** включать элементы удаления для удалённых SA, поскольку это будет приводить к дублированию удаления и может (теоретически) удалить ненужную SA.

Узлу **следует** рассматривать полузакрытые соединения, как аномалию, и при их сохранении делать запись в журнал аудита. Отметим, что в этой спецификации не задаётся никаких временных интервалов, поэтому конечные точки сами устанавливают время ожидания. Узел **может** отвергнуть приём входящих данных через полузакрытое соединение, но **недопустимо** закрывать его в одностороннем порядке и после этого снова использовать SPI. Если состояние соединения в достаточной степени беспорядочным, узел **может** закрыть IKE_SA; в этом случае он неявно закрывает все согласованные в нем SA. После этого узел может заново создать требуемые SA на базе новой IKE_SA.

Обмен INFORMATIONAL определяется следующим образом:

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

Обработка информационного обмена определяется включёнными в него элементами данных.

1.5. Информационные сообщения вне IKE_SA

Если зашифрованный пакет IKE приходит в порт 500 или 4500 с нераспознанным SPI, причиной этого может быть недавний сбой принимающего узла и потеря информации о состоянии, тот или иной системный отказ или атака. Если принимающий узел имеет активную IKE_SA для IP-адреса, с которого пришел пакет, он **может** передать уведомление о странном пакете через IKE_SA, используя обмен INFORMATIONAL. Если узел не имеет такой IKE_SA, он **может** отправить информационное сообщение без криптографической защиты по адресу отправителя пакета. Такое сообщение не является частью информационного обмена и для принявшего это сообщение узла **недопустимо** отвечать на него. Такой ответ может привести к возникновению петли при обмене сообщениями.

2. Детали и вариации протокола IKE

IKE обычно слушает и передаёт дейтаграммы UDP через порт 500, хотя сообщения IKE принимаются также через порт UDP 4500 с использованием слегка отличающегося формата (см. параграф 2.23). Поскольку протокол UDP использует дейтаграммы (транспорт без гарантии доставки), IKE включает определение процедуры восстановления при ошибках передачи, включая потерю и повторное использование пакетов, а также приём поддельных пакетов. Протокол IKE рассчитан на работу в условиях, когда (1) по крайней мере один из серии переданных повторно пакетов достигает получателя до завершения времени ожидания и (2) канал не переполнен обманными и повторными пакетами так, что это ведёт к нехватке ресурсов сети или производительности CPU[1] на одной из конечных точек. Даже при невыполнении этих минимальных требований IKE будет прерывать работу «чисто» (как при обрыве сети).

Хотя сообщения IKEv2 должны быть короткими, они содержат структуры данных без жёстко заданной верхней границы размера (в частности, сертификаты X.509), а сам протокол IKEv2 не включает механизма фрагментирования больших сообщений. Протокол IP определяет механизм фрагментирования слишком больших дейтаграмм UDP, но максимальный поддерживаемый размер может зависеть от реализации. Более того, использование фрагментации IP открывает реализации для атак на службы (DoS[2]) [KPS03]. Кроме того, некоторые реализации NAT и/или межсетевых экранов могут блокировать фрагменты IP.

Все реализации IKEv2 **должны** быть способны передавать, принимать и обрабатывать сообщения IKE размером до 1280 байтов и **следует** также обеспечивать возможность передачи, приёма и обработки сообщений размеров до 3000 байтов. Реализациям IKEv2 **следует** принимать во внимание максимальный размер поддерживаемых сообщений UDP и **можно** укорачивать сообщения, убирая из них некоторые предлагаемые сертификаты и криптографические наборы, если это позволит сохранить размер сообщения ниже максимума. Использование форматов «Hash and URL[3]» там, где это возможно, позволит избежать большинства проблем. В реализациях и конфигурационных параметрах следует принимать во внимание, что в тех случаях, когда поиск URL становится возможным только после организации IPsec SA, проблемы рекурсии могут воспрепятствовать применению упомянутого метода.

[1] Central Processor Unit – центральный процессор. *Прим. перев.*
[2] Denial of service attack – атака на отказ службы.
[3] Хэш и указатель ресурса.

2.1. Использование таймеров повтора передачи

Все сообщения в IKE существуют попарно - запрос и отклик. Организация IKE_SA обычно включает две пары запрос-отклик. После организации IKE_SA любая из сторон защищённой связи может в любой момент инициировать запрос и в каждый момент времени «налету» может находиться множество запросов и откликов. Но каждое сообщение помечается, как запрос или отклик и для каждой пары запрос-отклик одна из сторон является инициатором, а другая - ответчиком.

Для каждой пары сообщений IKE инициатор несёт ответственность за повтор сообщения при тайм-аутах. Отвечающая сторона никогда не **должна** передавать отклик повторно без получения повторного запроса. В этом случае ответчик **должен** передать отклик на повторный запрос, не повторяя связанных с его обработкой действий. Отвечающий **должен** помнить каждый отклик до момента получения запроса порядковым номером, превышающим номер переданного запроса плюс размер окна (см. параграф 2.3).

IKE является протоколом с гарантированной доставкой в том смысле, что инициатор **должен** повторять передачу запроса, пока на него не будет получен отклик **или** не будет принято решение об отказе защищённой связи IKE с отбрасыванием всей информации о состояниях, связанной с данной IKE_SA и всеми CHILD_SA, согласованными в этой IKE_SA.

2.2. Использование порядковых номеров для Message ID

Каждое сообщение IKE содержит идентификатор Message ID, который является частью фиксированного заголовка. Этот идентификатор используется для поиска соответствия между запросами и откликами, а также для аутентификации повтора сообщений.

Message ID представляет собой 32-битовое число, которое принимает нулевое значение при передаче в IKE первого запроса в каждом направлении. Начальные сообщения при организации IKE_SA всегда будут иметь номера 0 и 1. Каждая конечная точка в IKE SA поддерживает два «текущих» значения Message ID - следующее, которое будет использоваться при инициировании запроса и следующее, которое она ожидает получить в запросе от другой стороны. Значения счётчиков инкрементируются при генерации и получении запросов, соответственно. Отклик всегда содержит значение Message ID из соответствующего запроса. Это означает, что после начального обмена каждое целое значение n может появляться в качестве идентификатора 4 разных сообщений - n-ого запроса от исходного инициатора IKE, соответствующего ему отклика, n-ого запроса от исходного ответчика IKE и соответствующего ему отклика. Если стороны делают разное число запросов, значения Message ID в разных направлениях могут существенно различаться. Однако здесь не возникает неоднозначности в сообщениях, поскольку биты инициатора (I) и ответчика (R) в заголовке сообщения показывают, которым из четырёх возможных сообщений является данное.

Отметим, что значение Message ID шифруется и для него обеспечивается защита целостности для предотвращения повторного использования сообщений. В маловероятной ситуации, когда значение Message ID достигает предела 32-битового числа, **требуется** закрыть IKE_SA. Замена ключей IKE_SA ведёт к сбросу значений идентификаторов.

2.3. Размер окна для перекрывающихся запросов

Для достижения максимальной производительности IKE конечная точка IKE **может** вводить множество запросов до получения ответа на какой-либо из них, если другая точка указала свою способность обрабатывать множественные запросы. Для простоты реализация IKE **может** выбрать обработку запросов строго в порядке их подачи и/или подачу следующего запроса только после получения ответа на предыдущий. Необходимо ввести некоторые правила для обеспечения взаимодействия между реализациями, использующими разную стратегию.

После организации IKE_SA любая из сторон может инициировать один или множество запросов. Эти запросы могут могут проходить через сеть с изменением порядка следования. Конечная точка IKE **должна** быть готова к восприятию запроса в то время, когда ещё не завершена обработка предыдущего запроса, чтобы избежать возникновения тупиковых ситуаций. Конечной точке IKE **следует** быть готовой к восприятию и обработке множества запросов при незавершённой обработке имеющихся запросов.

Конечная точка IKE **должна** ждать отклика на каждое из своих сообщений до передачи следующего сообщения, если она не получила от партнёра уведомление SET_WINDOW_SIZE о готовности партнёра поддерживать состояние для множества обрабатываемых запросов с целью повышения производительности.

Для конечной точки IKE **недопустимо** выходить за пределы заявленного партнёром размера окна при передаче запросов IKE. Иными словами, если отвечающая сторона заявляет для своего окна размер N, инициатору для того, чтобы отправить запрос X, **требуется** необходимо дождаться откликов на все запросы, вплоть до X-N. Конечная точка IKE **должна** хранить копию (или обеспечивать точное воспроизведение) каждого переданного ею запроса, пока на этот запрос не был получен отклик. Конечная точка IKE **должна** хранить копию (или обеспечивать точное воспроизведение) предыдущих откликов в количестве, равном объявленному ею размеру окна, на случай потери отклика и получения от инициатора повторного запроса.

Конечной точке IKE, поддерживающей окно размером больше 1, **следует** обеспечивать возможность обработки входящих запросов, доставленных с нарушением порядка, для повышения пропускной способности в случаях разупорядочения пакетов или возникновения отказов в сети.

2.4. Синхронизация состояний и время ожидания для соединений

Конечным точкам IKE разрешается в любой момент забывать все свои состояния, связанные с IKE_SA и набором соответствующих CHILD_SA. Это сделано для обеспечения устойчивости к авариям и перезапускам конечных точек. Важно, чтобы при аварии или повторной инициализации состояния конечной точки другая сторона детектировала такие события и прекращала бы расход полосы сети на передачу пакетом через отброшенную SA, которые будут уходить в «чёрную дыру».

Поскольку протокол IKE был разработан с учётом возможности атак на отказ служб (DoS) из сети, для конечной точки **недопустимо** констатировать отказ другой конечной точки на основе какой-либо маршрутной информации (например, сообщений ICMP) или сообщений IKE, приходящих без криптографической защиты (например, сообщений Notify о неизвестных SPI). Конечная точка **должна** констатировать отказ другой конечной точки только в случаях

повторяющихся в течение всего периода ожидания отказах (отсутствии ответов) при попытках контакта с этой точкой или при получении криптографически защищённого уведомления INITIAL_CONTACT для другой IKE_SA с такой же аутентификацией. Конечной точки на основании соответствующей маршрутной информации и инициировать запрос для проверки жизнеспособности другой точки. Для такой проверки в IKE предусмотрены пустые сообщения INFORMATIONAL, которые (подобно всем запросам IKE) требуют подтверждения (отметим, что в контексте IKE_SA «пустое» сообщение представляет собой заголовок, за которым следует поле Encrypted, не содержащее данных). Если от другой стороны недавно было получено криптографически защищённое соединение, незащищённые уведомления **можно** игнорировать. Реализации **должны** ограничивать частоту операций, выполняемых на основе незащищённых соединений.

Число попыток и продолжительность времени ожидания не задаются данной спецификацией, поскольку они не оказывают влияния на взаимодействие. Предлагается повторять передачу сообщения по крайней мере дюжину раз в течение периода по крайней мере в несколько минут прежде, чем отказаться от SA, однако в разных средах эти параметры могут различаться. Для предотвращения возможных перегрузок период повтора передачи сообщений **должен** возрастать экспоненциально. Если на всех SA, связанных с IKE_SA, присутствовал только исходящий трафик, важно убедиться в жизненности другой конечной точки для предотвращения «чёрных дыр». Если в течение некоторого времени не было получено криптографически защищённых сообщений в IKE_SA или любой из дочерних CHILD_SA, система должна проверить жизненность удалённой точки для предотвращения передачи сообщений «мёртвому» партнёру. Получение свежего, криптографически защищённого сообщения в IKE_SA или любой из дочерних CHILD_SA гарантирует жизненность IKE_SA и всех дочерних CHILD_SA. Отметим, что это вносит требования к обработке отказов конечных точек IKE. Для реализации **недопустимо** продолжать передачу в любую SA, если тот или иной отказ не позволяет принимать сообщения на всех связанных SA. Если возможен отказ одной CHILD_SA независимо от других без возможности для IKE_SA передачи сообщения Delete, для таких SA **должны** согласовываться раздельные IKE_SA.

Существуют DoS-атаки на инициатора IKE_SA, которых можно избежать в случае применения инициатором соответствующих мер. Поскольку два первых сообщения при организации SA не защищаются криптографически, атакующий может ответить на сообщения инициатора раньше, чем вызываемая сторона, и сорвать организацию соединения. Для предотвращения этого инициатор **может** выбрать восприятие множества откликов на своё первое сообщение, трактуя их, как потенциально легитимные, а потом отбросить все некорректные полуоткрытые соединения, когда будет получен корректный, криптографически защищённый отклик на любой из его запросов. После получения криптографически корректного отклика все последующие отклики[1] следует игнорировать, независимо от их криптографической корректности.

Отметим, что с приведёнными правилами не возникает необходимости согласования срока жизни SA. Если IKE предполагает, что партнёр не работает на основе повторяющегося отсутствия подтверждений для сообщения IKE, тогда IKE SA и все дочерние CHILD_SA, организованные в данной IKE_SA, удаляются.

Конечная точка IKE может в любой момент удалить неактивную CHILD_SA в целях освобождения ресурсов, используемых для поддержки состояния этих связей. Если конечная точка IKE принимает решение об удалении CHILD_SA, она **должна** передать другой стороне элемент Delete для уведомления об удалении. Это **может** быть похоже на тайм-аут для IKE_SA. Закрытие IKE_SA ведёт к неявному закрытию всех связанных CHILD_SA. В этом случае конечной точке IKE **следует** передать элемент Delete, показывающий удаление IKE_SA.

2.5. Номера версий и совместимость

В этом документе описывается версия 2.0 протокола IKE - старшая часть версии имеет номер 2, а младшая - 0. Очевидно, что некоторые реализации захотят поддерживать версии 1.0 и 2.0, а в будущем и другие версии.

Старшую часть номера версии следует менять только в тех случаях, когда формат пакетов или требуемые действия меняются столь существенно, что узлы старой версии не смогут взаимодействовать напрямую с узлами более новой версии, если они будут просто игнорировать поля непонятные новой версии и выполнять действия, заданные в старой спецификации. Младшая часть номера версии показывает новые возможности и **должна** игнорироваться узлом с меньшим значением младшей части номера версии, но использоваться для информации с большим значением младшего номера версии. Например, младшая часть может показывать возможность обработки новых уведомляющих сообщений. Узел с большим значением младшей части номера будет просто отмечать неспособность своего корреспондента понимать такие сообщения и, следовательно, передавать их.

Если конечная точка получает сообщение с большим (чем у неё) значением старшей части номера версии, она **должна** отбросить такое сообщение; **следует** также передать в ответ неаутентифицированное уведомление, содержащее поддерживаемое значение старшей части номера версии. Если конечная точка поддерживает версию со старшей частью n и m, она **должна** также поддерживать все версии между n и m. Если такая точка получает сообщение поддерживаемой версии, она **должна** отвечать сообщением той же версии. Для предотвращения ситуаций, когда пара узлов использует старшую часть номера версии меньше максимально поддерживаемого обоими узлами номера, в IKE используется флаг, показывающий, что узел способен поддерживать больший номер версии.

Таким образом, старшая часть номера версии в заголовке IKE показывает номер версии для данного сообщения, а не номер версии, поддерживаемой отправителем. Если инициатор способен поддерживать версии n, n+1 и n+2, а отвечающая сторона поддерживает версии n и n+1, они согласуют использование версии n+1, а инициатор будет устанавливать флаг способности поддерживать более высокую версию. Если узлы по ошибке (или в результате активной атаки) согласуют использование версии n, тогда оба узла будут указывать поддержку более высокой версии. В этом случае они **должны** разорвать соединение и организовать его заново с использованием версии n+1.

Отметим, что IKEv1 не следует этим правилам, поскольку в этой версии протокола просто не может быть указана поддержка более высокой версии. Поэтому в активной атаке может просто использоваться попытка вынудить пару узлов v2 работать на основе v1. Когда узел, поддерживающий v2, согласует работу на основе v1, ему **следует** отмечать этот факт в системном журнале.

Для обеспечения совместимости с более новыми версиями во всех резервных полях реализации версии 2.0 **должны** устанавливать значение 0, а при получении сообщений такие реализации **должны** игнорировать содержимое резервных полей (будьте консервативными при передаче и либеральными при приёме). В результате новые версии

[1] На тот же запрос. *Прим. перев.*

протокола смогут использовать резервные поля так, что их значения будут игнорироваться реализациями, не понимающими таких полей. Аналогично, типы данных, которые не определены, являются резервными[1].

IKEv2 добавляет флаг «критичности» (critical) к каждому заголовку данных для более гибкой совместимости с грядущими версиями. Если флаг critical установлен, а тип данных нераспознан, сообщение **должно** быть отвергнуто, а отклик на запрос IKE, включающий эти данные, **должен** включать элемент Notify UNSUPPORTED_CRITICAL_PAYLOAD, показывающий приём нераспознанных критичных данных. Если флаг критичности не установлен, нераспознанный элемент **должен** игнорироваться.

Хотя в будущем могут добавляться новые элементы данных, которые будут появляться вперемешку с определёнными в этой спецификации полями, реализации **должны** передавать определённые в этой спецификации элементы в том порядке, который показан на рисунках раздела 2 и реализациям **следует** отвергать некорректные сообщения с другим порядком элементов данных.

2.6. Cookie

Термин cookie, введённый Karn и Simpson [RFC2522] в Photuris - раннем варианте системы управления ключами IPsec, продолжает использоваться до настоящего времени. Фиксированный заголовок протокола управления ключами и защищёнными связями Internet (ISAKMP) [MSST98] включает два восьмиоктетных поля cookies и этот синтаксис используется в IKEv1 и IKEv2, хотя в последнем эти поля обозначаются, как IKE SPI, и имеется новое отдельное поле в элементе Notify, которое сохраняет значение cookie.

Изначально два восьмиоктетных поля в заголовке использовались в качестве идентификаторов соединений в начальных пакетах IKE. Каждая из конечных точек выбирает одно или два значения SPI, которые **следует** выбирать так, чтобы они однозначно идентифицировали IKE_SA. Нулевое значение SPI является специальным и показывает, что удалённое значение SPI отправителю ещё не известно.

В отличие от ESP и AH, где только значение SPI для получателя появляется в заголовке пакета, в IKE каждое сообщение содержит также SPI отправителя. Поскольку значение SPI, выбранное исходным инициатором IKE_SA, всегда передаётся первым, конечная точка со множеством открытых IKE_SA, которая хочет найти подходящую IKE_SA по выбранному для неё значению SPI, должна просматривать значение флага I (инициатор) в заголовке для определения где искать - в первой или второй группе из 8 октетов.

В первом сообщении начального обмена IKE инициатор не знает значение SPI отвечающей стороны и будет, следовательно, помещать в это поле нулевое значение.

Возможной атакой на IKE является истощение ресурсов на хранение состояний и ресурсов CPU, когда объект атаки в лавинном режиме получает запросы на организацию сессий с подставных адресов IP. Воздействие таких атак можно снизить, если реализация отвечающей стороны по минимуму использует CPU и не фиксирует состояния SA до того, как узнает, что инициатор может получать пакеты по адресу, указанному в запросе. Для решения этой задачи ответчику **следует** при обнаружении большого числа полуоткрытых IKE_SA отвергать стартовые сообщения IKE, если они не содержат элемента Notify типа COOKIE. Взамен ответчику **следует** передавать в качестве отклика незащищённое сообщение IKE и включать в него COOKIE Notify с данными cookie для возврата. Инициатор, получивший такой отклик, **должен** повторить IKE_SA_INIT с элементом Notify типа COOKIE, содержащим предложенные ответчиком данные cookie в качестве первого элемента, сохраняя остальные элементы данных. Начальный обмен в таком случае будет иметь вид:

```
 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}
```

Два первых сообщения не оказывают влияния на состояние инициатора и ответчика за исключением обмена cookie. В частности, порядковые номера в четырёх первых сообщениях будут иметь нулевые значения, а в двух последних сообщениях приведённого примера номера будут иметь значение 1. «A» обозначает значение SPI, присвоенное инициатором, а «B» - значение, присвоенное ответчиком.

Реализации IKE **следует** поддерживать генерацию cookie ответчиком так, чтобы не требовалось сохранять какое-либо состояние для проверки корректности cookie при получении второго сообщения IKE_SA_INIT. Выбор алгоритма и используемый для cookie синтаксис не оказывают влияния на взаимодействие, поэтому не задаются данной спецификацией. Ниже приведён пример использования cookie конечной точкой для частичной защиты от DoS-атак.

Хорошим способом реализации такой защиты является установка ответчиком cookie по следующему алгоритму:

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

где <secret> - случайное значение, известное только ответчику и периодически сменяемое, | указывает конкатенацию. Значение <VersionIDofSecret> следует менять всякий раз при смене <secret>. Значение cookie может быть рассчитано заново при получении IKE_SA_INIT второй раз и полученное значение сравнивается со значением cookie в принятом сообщении. Если значения совпадают, ответчик знает, что значение cookie было создано после замены <secret> и значение IPi должно совпадать с адресом отправителя, полученным в первом сообщении. Включение SPIi в расчёт обеспечивает создание различных значений cookie для случаев, когда множество IKE_SA создаётся одновременно (предполагается, что инициаторы устанавливают уникальные значения SPIi). Включение в хэш значения Ni не позволяет атакующему, который увидел только сообщение 2, корректно подменить сообщение 3.

Если значение <secret> меняется в процессе инициализации соединения, сообщение IKE_SA_INIT может быть возвращено с отличным от текущего значением <VersionIDofSecret>. Ответчик в таком случае **может** отвергнуть сообщение, передавая другой отклик с новым значением cookie, или **может** сохранять старое значение <secret> на короткое время после его замены и принимать cookie, рассчитанные с использованием этого значения. Ответчику **не**

[1] В оригинале это предложение содержит ошибку, см. https://www.rfc-editor.org/errata_search.php?eid=2190. *Прим. перев.*

следует воспринимать cookie неограниченно долго после смены <secret>, поскольку это будет нарушать часть защиты от DoS-атак. Ответчику **следует** менять значение <secret> достаточно часто, особенно во время атак.

2.7. Согласование криптоалгоритма

Элемент данных типа SA показывает предложения в части выбора протоколов IPsec (IKE, ESP и/или AH) для SA, а также криптографических алгоритмов, связанных с каждым протоколом.

Элемент данных SA включает одно или несколько предложений. Каждое из предложений включает один или несколько протоколов (обычно один). Каждый протокол включает одно или несколько преобразований, каждое из которых задаёт криптографический алгоритм. Каждое преобразование может включать атрибуты (атрибуты нужны лишь в тех случаях, когда идентификатор преобразования не задаёт криптографический алгоритм полностью).

Такая иерархическая структура была разработана для эффективного представления предложений по выбору криптографических наборов, когда число поддерживаемых наборов велико, поскольку множество значений приемлемо для множества платформ. Отвечающая сторона **должна** выбрать один набор, который **может** быть любым подмножеством предложения в SA, выбранным в соответствии с приведёнными ниже правилами:

> Каждое предложение включает, по крайней мере, один протокол. Если предложение принимается элемент SA в отклике **должен** содержать те же протоколы в том же порядке, как они были указаны в предложении. Ответчик **должен** принять одно из предложений или отвергнуть все предложения и возвратить сообщение об ошибке (Например, если предложение включает протоколы ESP и AH и это предложение принимается, оба протокола ESP и AH **должны** восприниматься. Если ESP и AH включены в разные предложения, ответчик **должен** принять только один из этих протоколов[1]).
>
> Каждый предлагаемый протокол IPsec содержит, по крайней мере, одно преобразование. Каждое преобразование включает тип преобразования. Принимаемые криптографические наборы **должны** содержать в точности одно преобразование каждого типа, включённого в предложение. Например, если предложение ESP включает преобразования ENCR_3DES, ENCR_AES w/keysize 128, ENCR_AES с размером ключа 256, AUTH_HMAC_MD5 и AUTH_HMAC_SHA, принятый набор **должен** включать одно преобразование ENCR_ и одно преобразование AUTH_. Таким образом, возможно шесть комбинаций.

Поскольку инициатор передаёт своё значение Diffie-Hellman в сообщении IKE_SA_INIT, он должен угадать группу Diffie-Hellman, которую ответчик выберет из списка поддерживаемых групп. Если выбор инициатора окажется ошибочным, ответчик будет возвращать элемент данных Notify типа INVALID_KE_PAYLOAD, показывающий выбранную группу. В таком случае инициатор **должен** повторить запрос IKE_SA_INIT с указанием корректной группы Diffie-Hellman. Инициатор **должен** снова предложить полный набор допустимых криптографических наборов, поскольку сообщение с отказом было передано без аутентификации. Если этого не делать, активный атакующий сможет принудить конечные точки к выбору наиболее слабого криптографического набора из числа поддерживаемых обеими сторонами.

2.8. Смена ключей

Защищённые связи IKE, ESP и AH используют секретные ключи, которые **следует** применять в течение ограниченного периода времени для защиты ограниченного объёма данных. Эти требования ограничивают срок существования защищённых связей. Когда время жизни защищённой связи заканчивается, дальнейшее использование такой связи **недопустимо**. При необходимости **может** быть организована новая защищённая связь. Повторная организация защищённой связи при завершении срока существования последней называется сменой ключей (rekeying).

Для того, чтобы сохранить возможность создания реализаций IPsec с минимальным набором возможностей, смена ключей SA без повторной организации IKE_SA целиком является необязательной. Реализация **может** отвергать все запросы CREATE_CHILD_SA в IKE_SA. Если срок жизни SA закончился или близок к завершению и попытки смены ключей с использованием описанных здесь механизмов не дали положительного результата, реализация **должна** закрыть IKE_SA и все дочерние CHILD_SA, а после этого **может** организовать новые связи. Реализациям **следует** поддерживать замену ключей для SA, поскольку это обеспечивает повышение производительности и снижение числа пакетов, теряемых в переходный период.

Для смены ключей CHILD_SA в существующей IKE_SA создаётся новая, эквивалентная SA (см. параграф 2.17 ниже) и после создания старая связь удаляется. Для смены ключей IKE_SA создаётся новая, эквивалентная IKE_SA (см. параграф 2.18 ниже) с партнёром, который в старой IKE_SA совместно использовался в CREATE_CHILD_SA. Созданная таким путём IKE_SA наследует все CHILD_SA исходной in-place. Используется новая IKE_SA для всех управляющих сообщений, требуемых для поддержки CHILD_SA, созданных старой IKE_SA, после чего старая IKE_SA удаляется. Элемент данных Delete для удаления самой связи **должен** быть последним запросом, передаваемым через старую IKE_SA.

Ключи SA следует менять заранее, не дожидаясь, пока время жизни старых связей закончится и использовать их станет невозможно. Хотя между организацией новой SA и завершением срока жизни старой может остаться достаточно времени, трафик может быть переключен в новую SA.

Различие между IKEv1 и IKEv2 заключается в том, что времена жизни SA в IKEv1 согласовывались. В IKEv2 каждая из сторон SA отвечает за свою собственную политику в плане срока жизни SA и меняет ключи для SA по необходимости. Если стороны используют разные правила для срока жизни связей, сторона с меньшим сроком всегда будет той, которая вводит запрос на замену ключей. Если группа SA не активна в течение долгого времени и при отсутствии трафика SA не будут инициироваться, конечная точка может закрыть SA по истечении времени жизни, вместо смены ключей для этой связи. Так **следует** поступать в тех случаях, когда трафик через SA отсутствовал с момента предыдущей смены ключей.

Если политика обеих сторон в части времени жизни совпадает, они могут ввести запросы на смену ключей одновременно (это приведёт к созданию избыточных SA). Для снижения вероятности такого события **следует** использовать вариации периода смены ключей (при получении уведомления о необходимости замены ключей вносится случайная задержка).

[1] Или отказаться от обоих. *Прим. перев.*

Такая форма смены ключей может приводить к временному существованию множества похожих SA между одной парой узлов. При наличии двух SA, подходящих для получения пакетов, узел **должен** воспринимать входящие пакеты из обеих SA. Если при таком конфликте создаются избыточные SA, связь, имеющую наименьшее из четырёх использованных в этих двух обмена значений nonce, **следует** закрыть (конечной точке, которая создала эту связь).

Отметим, что существование параллельных SA с одинаковым трафиком между парой конечных точек разрешено в IKEv2 осознанно. Одной из причин этого является поддержка различий в качестве обслуживания трафика (QoS[1]) между SA (см. [RFC2474], [RFC2475] и параграф 4.1 в [RFC2983]). Следовательно, в отличие от IKEv1, комбинация конечных точек и селекторов трафика может не быть уникальным идентификатором SA между парой точек, поэтому принятое при смене ключей в IKEv1 эвристическое удаление SA на основе совпадения селекторов трафика **не следует** использовать.

Узлу, который инициировал SA при досрочной замене ключей, **следует** удалить заменённую SA после создания новой.

Существуют интервалы времени (в частности, в присутствии потерь пакетов), когда конечные точки могут по разному трактовать состояние SA. Отвечающая на CREATE_CHILD_SA сторона **должна** быть готова к восприятию сообщений через SA до передачи своего отклика на запрос создания связи, поэтому здесь для инициатора не возникает неоднозначности. Инициатор **может** начать передачу в SA сразу после обработки отклика. Инициатор, однако, не может принимать через новую SA до получения и обработки отклика на свой запрос CREATE_CHILD_SA. Как, в таком случае, ответчик может узнать о возможности начать передачу в новую SA?

С точки зрения технической корректности и взаимодействия ответчик может начать передачу через SA, как только он отправит свой отклик на запрос CREATE_CHILD_SA. Однако в некоторых случаях это может привести к неоправданному отбрасыванию пакетов, поэтому реализация **может** принять решение о задержке такой передачи.

Ответчик может быть уверен в том, что инициатор готов получать сообщения через SA, если (1) он получил криптографически корректное сообщение через новую SA или (2) новая SA создана при замене ключей для существующей SA и был получен запрос IKE на закрытие заменённой SA. При смене ключей SA ответчику **следует** продолжать передачу сообщений через старую SA, пока не будет выполнено какое-либо из приведённых выше условий. При создании новой SA ответчик может отложить передачу сообщений через неё до получения сообщения через неё или завершения времени ожидания. Если инициатор получает сообщение в SA, для которой он ещё не получил отклика на свой запрос CREATE_CHILD_SA, ему **следует** интерпретировать это событие, как потерю пакетов, и передать запрос CREATE_CHILD_SA повторно. Инициатор **может** передать бутафорское (dummy) сообщение через новую SA, если у него нет сообщений в очереди, чтобы уведомить ответчика о своей готовности к приёму сообщений.

2.9. Согласование селекторов трафика

Когда полученный поддерживающей RFC4301 системой IPsec пакет IP соответствует «селектору защиты» в соответствии с SPD[2], подсистема **должна** защитить пакет с использованием IPsec. Если SA ещё не существует, её создание является задачей IKE. Поддержка SPD в системе выходит за пределы IKE (см. [PFKEY] в качестве примера протокола), хотя некоторые реализации могут обновлять свои SPD в контакте с работающим IKE (см., для примера, сценарий 1.1.3).

Элементы данных TS[3] позволяют конечным точкам обмениваться некой информацией из SPD со своими партнёрами. Элементы TS задают критерии выбора пакетов, которые будут передаваться через вновь созданную SA. Это может служить проверкой согласованности в некоторых сценариях для контроля непротиворечивости SPD. В других случаях это ведёт к динамическому обновлению SPD.

В каждом из сообщений обмена, создающего пару CHILD_SA, появляется два элемента TS. Каждый из TS содержит один или множество селекторов трафика. Каждый селектор состоит из диапазона адресов (IPv4 или IPv6), диапазона портов и идентификатора протокола IP. В поддержку сценария, описанного в параграфе 1.1.3, инициатор может запросить у отвечающей стороны выделение адреса IP с его кратким описанием (что это?).

IKEv2 позволяет ответчику выбрать подмножество селекторов трафика, предложенных инициатором. Это может происходить в тех случаях, когда конфигурация двух конечных точек была изменена, но только одна сторона получила новую информацию. Поскольку обе конечных точки могут настраиваться разными людьми, может возникать период несовместимости даже при отсутствии ошибок. Это также разрешает заведомо разные конфигурации, когда одна сторона настроена на туннелирование всех адресов и обновление списка зависит от другой стороны.

Первый из двух элементов TS называют TSi[4], а второй - TSr[5]. TSi задаёт адрес отправителя трафика, пересылаемого от инициатора (или адрес получателя трафика, пересылаемого инициатору) пары CHILD_SA. TSr указывает адрес получателя трафика, пересылаемого ответчика (или адрес отправителя трафика, пересылаемого от ответчика) пары CHILD_SA. Например, если исходный инициатор запрашивает создание пары CHILD_SA и хочет туннелировать весь трафик из подсети 192.0.1.* на своей стороне в подсеть 192.0.2.*[6] на стороне ответчика, инициатор будет включать один селектор трафика в каждый элемент TS. TSi будет задавать диапазон адресов 192.0.1.0 - 192.0.1.255, а TSr - диапазон 192.0.2.0 - 192.0.2.255. Предположим, что предложение будет принято ответчиком - тогда он будет передавать обратно идентичные элементы TS.

Ответчику разрешено сужать выбор путём выделения подмножества трафика (например, путём исключения или диапазона для одного или множества элементов набора дескрипторов трафика), но выбранное подмножество не должно быть пустым.

Политика отвечающей стороны может содержать множество меньших диапазонов, охватываемое предложенным инициатором селектором трафика, причём политика требует передачи трафика для этих диапазонов через разные SA.

---

[1]Quality of service.
[2]Security Policy Database - база данных о правилах защиты.
[3]Traffic Selector - селектор трафика.
[4]Traffic Selector-initiator - селектор трафика от инициатора.
[5]Traffic Selector-responder - селектор трафика отвечающей стороны.
[6]Блок адресов IP 192.0.2.* зарезервирован для использования в примерах для RFC и аналогичных документов. Поскольку в данном документе нужны два таких диапазона, используется также блок адресов 192.0.1.*. Не следует путать эти блоки с реальными адресами.

Продолжим использование приведённых выше для примера адресов. Ответчик может иметь политику, которая позволяет туннелировать эти адреса в направлении инициатора и от него, но может требовать, чтобы для каждой пары адресов независимо согласовывалась CHILD_SA. Если инициатор генерирует свой запрос в ответ на пакет с адреса 192.0.1.43, направленный по адресу 192.0.2.123, у ответчика не будет способа определить, какую пару адресов следует включить в этот туннель и он будет пытаться угадать или отбросить запрос со статусом SINGLE_PAIR_REQUIRED.

Чтобы позволить ответчику выбрать подходящий диапазон в том случае, когда инициатор запрашивает SA в результате получения пакета данных, инициатору **следует** включить в качестве первого селектора трафика в каждом элементе Tsi и TSr очень специфичный селектор трафика, включающий адреса в пакете, вызвавшем запрос. В нашем примере инициатор будет включать в TSi два селектора трафика - первый будет содержать диапазон адресов 192.0.1.43 - 192.0.1.43, а также порт отправителя и протокол IP из пакета, а второй - диапазон 192.0.1.0 - 192.0.1.255 со всеми портами и протоколами. Инициатор будет также включать два селектора трафика в TSr.

Если политика отвечающей стороны не позволяет принять весь набор селекторов трафика из запроса инициатора, но позволяет принять первый селектор TSi и TSr, ответчик **должен** сузить селекторы трафика до подмножества, включающего первый выбор инициатора. В приведённом примере ответчик может возвратить TSi 192.0.1.43 - 192.0.1.43 с поддержкой всех портов и протоколов IP.

Если инициатор создаёт CHILD_SA пару не в ответ на получение пакета, а, например, при старте, может не быть предпочтений в части адресов для начального туннеля. В таких случаях первые значения TSi и TSr **могут** задавать диапазоны, а не конкретные адреса и ответчик выбирает подходящее подмножество указанных инициатором TSi и TSr. Если ответчика устраивает несколько подмножеств, которые нельзя объединить, ответчик **должен** принять некое подмножество и **может** включить в сообщение элемент Notify типа ADDITIONAL_TS_POSSIBLE для индикации инициатору возможности повтора. Такая ситуация возникает только в тех случаях, когда конфигурации инициатора и ответчика различаются. Если инициатор и ответчик согласовали гранулярность туннелей, инициатор никогда не будет запрашивать более широкий туннель, нежели приемлет отвечающая сторона. Такие расхождения в конфигурационных параметрах **следует** заносить в системный журнал ошибок.

2.10. Элементы nonce

Каждое сообщение IKE_SA_INIT содержит nonce. Эти nonce используются в качестве входной информации для криптографических функций. Запросы CREATE_CHILD_SA и отклики CREATE_CHILD_SA также включают nonce. Эти nonce используются для повышения эффективности метода, используемого при получении ключей для CHILD_SA, обеспечения достаточно высокого уровня случайности для псевдослучайных битов, получаемых из ключа Diffie-Hellman. Элементы nonce, используемые в IKEv2, **должны** выбираться случайным образом, **должны** иметь размер не менее 128 битов и не менее половины размера ключа согласованной prf[1]. При использовании некого общего источника случайных чисел для ключей и nonce нужно быть осторожными, чтобы использование nonce не привело к компрометации ключей.

2.11. Использование адресов и портов

IKE работает по протоколу UDP через порты 500 и 4500, неявно устанавливая связи ESP и AH для тех же адресов IP. Адреса IP и номера портов во внешнем заголовке не защищаются криптографически и протокол IKE может работать даже через устройства трансляции адресов (NAT). Реализация **должна** воспринимать входящие запросы даже из портов с номерами, отличными от 500 или 4500 и **должна** отвечать на адрес и порт, с которых был получен запрос. Реализация **должна** указать в отклике адрес и порт, через которые был принят запрос, в качестве адреса и порта отправителя. Функции IKE идентичны для протоколов IPv4 и IPv6.

2.12. Многократное использование экспоненциалов Diffie-Hellman

Для обеспечения высокого уровня защиты[2] IKE генерирует короткоживущие ключи с использованием обмена Diffie-Hellman. Это означает, что после закрытия соединения соответствующие ключи забываются. Если кто-либо смог записать всю переданную через соединение информацию и получил доступ к долгосрочным ключам обеих сторон соединения, он не сможет восстановить ключи, которые использовались для соединения без перебора[3] всего пространства сеансовых ключей.

В соответствии с требованиями по обеспечению высокого уровня защиты каждая конечная точка при закрытии соединения **должна** забыть не только использованные в соединении ключи, но и любую информацию, которая применялась для расчёта этих ключей. В частности **необходимо** забыть секреты, использованные в расчётах Diffie-Hellman и любые состояния, которые могли бы сохраниться в состоянии генератора псевдослучайных чисел, используемого для пересчёта секретов Diffie-Hellman.

Поскольку расчёт экспоненциалов Diffie-Hellman требует значительных вычислительных ресурсов, конечные точки могут получать преимущества в результате использования этих экспоненциалов для организации множества соединений. Существует несколько вариантов обоснованной стратегии многократного использования экспоненциалов. Конечная точка выбирает новый экспоненциал периодически, хотя это может приводить к снижению уровня защиты. Другим вариантом является сохранение информации об использованных для каждого соединения экспоненциалов и удалять связанную с экспоненциалом информацию только после закрытия соответствующего соединения. Это позволяет использовать экспоненциалы многократно без снижения уровня защиты, но за счёт поддержки дополнительной информации о состоянии.

Решение вопроса о многократном использовании экспоненциалов Diffie-Hellman и способе такого использования является частным и не влияет на взаимодействие. Реализация, использующая экспоненциалы многократно, **может** принять решение о сохранении экспоненциалов, использованных другой точкой в предшествующих обменах, избегая второй половины расчётов при использовании этих экспоненциалов.

---

[1] Pseudo-random function - псевдослучайная функция - один из криптоалгоритмов, согласуемых в обмене IKE.
[2] В оригинале «perfect forward secrecy». *Прим. перев.*
[3] Brute force - подбор ключей методом «грубой силы».

2.13. Материал для генерации ключей

В контексте IKE_SA согласуются четыре криптографических алгоритма - алгоритм шифрования, алгоритм защиты целостности, группа Diffie-Hellman и псевдослучайная функция (prf). Последняя используется при создании ключевого материала для всех криптографических алгоритмов, используемых как в IKE_SA, так и в дочерних CHILD_SA.

Мы полагаем, что каждый алгоритм шифрования и защиты целостности использует ключ фиксированного размера и случайно выбранное значение фиксированного размера может служить подходящим ключом. Для алгоритмов, которые воспринимают ключи переменного размера, **должен** указываться фиксированный размер ключа в процессе согласования криптографических преобразований. Для алгоритмов, в которых не все значения являются допустимыми ключами (например, DES или 3DES с чётностью ключа) криптографическим преобразованием **должен** задаваться алгоритм создания ключей из произвольных значений. Для функций защиты целостности на основе кода HMAC[1] фиксированным размером ключа является размер вывода нижележащей функции хэширования. Когда функций prf принимает ключ переменной длины и данные переменной длины, давая результат фиксированного размера (например, при использовании HMAC), применяются формулы из этого документа. Когда ключ для prf имеет фиксированный размер, представленные в качестве ключа данные усекаются или дополняются нулями, если приведённая ниже формула не задаёт специальной обработки.

Ключевой материал всегда производится, как выход согласованного алгоритма prf. Поскольку количество требуемого ключевого материала может быть больше, чем размер вывода алгоритма prf, мы будем использовать prf итеративно. Обозначим prf+ функцию, которая даёт на выходе псевдослучайный поток на основе входной информации prf в соответствии с приведёнными ниже правилами (| обозначает 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)
```

и т. д., пока не будет достаточно материала для расчёта всех требуемых ключей. Ключи берутся из выходной строки без учёта границ (например, если нужен 256-битовый ключ AES[2] и 160-битовый ключ HMAC, а функция prf даёт на выходе 160 битов, ключ AES будет взят из T1 и начальной части T2, а ключ HMAC возьмёт остаток T2 и начало T3).

Константа, добавляемая в конец каждой строки на входе prf, представляет собой один октет. Использование функции prf+ в данном документе не выходит за пределы 255-кратного увеличения размера результата prf.

2.14. Генерация ключевого материала для IKE_SA

Для расчёта разделяемых ключей сначала вычисляется значение SKEYSEED на основе nonce из обмена IKE_SA_INIT и разделяемого секрета Diffie-Hellman созданного при этом обмене. Значение SKEYSEED используется для расчёта семи других секретов - SK_d применяется при создании новых ключей для CHILD_SA, создаваемых в данной IKE_SA; SK_ai и SK_ar применяются в качестве ключа алгоритма защиты целостности для аутентификации компонент сообщений в последующих обменах; SK_ei и SK_er применяются для шифрования (и дешифровки) всех последующих обменов; SK_pi и SK_pr применяются при генерации элемента данных AUTH.

SKEYSEED и производные от него ключи рассчитываются 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)
```

(левая часть второго уравнения показывает, что значения SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi и SK_pr берутся в указанном порядке из битов результата prf+). Параметр g^ir является разделяемым секретом из краткосрочного[3] обмена Diffie-Hellman. Значение g^ir представляется строкой октетов в формате big endian[4] с дополнением при необходимости нулями для выполнения требований по размеру модуля. Ni и Nr - значения элементов nonce, извлечённые из любых заголовков. Если согласованная функция prf принимает ключ фиксированного размера, а суммарная длина Ni и Nr превышает нужное значения, берётся половина (первая) битов из Ni и половина (первая) битов из Nr.

Для двух направления потока трафика используются разные ключи. Ключи, служащие для защиты сообщений от исходного инициатора, обозначаются SK_ai и SK_ei. Ключи, служащие для защиты сообщений в другом направлении, обозначаются SK_ar и SK_er. Каждый алгоритм принимает фиксированное число битов ключевого материала, заданное как часть алгоритма. Для алгоритмов защиты целостности на основе хэш-функций размер ключа всегда равен размеры результата нижележащей хэш-функции.

2.15. Аутентификация IKE_SA

Если не используется расширяемая аутентификация (см. параграф 2.16), партнёры аутентифицируют себя посредством подписи (или MAC[5] с использованием разделяемого секрета в качестве ключа) для блока данных. Для ответчика подписываемые данные начинаются с первого октета первого SPI в заголовке второго сообщения и заканчиваются последним октетом последнего элемента данных во втором сообщении. В конце к этому добавляется (с целью расчёта подписи) значение nonce Ni от инициатора (просто значение, а не содержащий его элемент данных) и значение prf(SK_pr,IDr'), где IDr' - элемент ID ответчика без фиксированного заголовка. Отметим, что ни одно из значений Ni и prf(SK_pr,IDr') не передаётся. Подобно этому инициатор подписывает первое сообщение, начиная с первого октета первого SPI в заголовке и заканчивая последним октетом последнего элемента данных. В конце к этому добавляется (для расчёта подписи) значение nonce Nr от ответчика и значение prf(SK_pi,IDi'). В приведённых выше расчётах IDi' и IDr' являются полными элементами данных ID без фиксированного заголовка. Для защиты обмена критично наличие подписи каждой стороны для nonce другой стороны.

---

[1] Hashed Message Authentication Code - код аутентификации хэшированных сообщений.
[2] Advanced Encryption Standard - улучшенный стандарт шифрования.
[3] В оригинале используется термин «ephemeral» (эфемерный). *Прим. перев.*
[4] Сначала старший байт. Такой порядок называют также сетевым (network byte order). *Прим. перев.*
[5] Message Authentication Code - код аутентификации сообщения. *Прим. перев.*

Отметим, что подписываются все элементы данных, включая и те, которые не определены в этом документе. Если первое сообщение в обмене передаётся дважды (второй раз с cookie ответчика и/или другой группой Diffie-Hellman), подписывается вторая версия сообщения.

В дополнение к сказанному сообщения 3 и 4 **могут** включать сертификат или цепочку сертификатов, обеспечивающие очевидность того, что использованные для расчёта цифровой подписи ключ относится к имени в элементе данных ID. Сигнатура или MAC будет рассчитываться с использованием алгоритмов, диктуемых типом ключа, используемого подписывающим, и задаётся полем Auth Method в элементе данных Authentication. Здесь не задаётся использования одного криптографического алгоритма для инициатора и ответчика. Выбор криптографического алгоритма зависит от типа ключа, который имеет каждая из сторон. В частности, инициатор может использовать разделяемый ключ, а ответчик может иметь открытый ключ подписи и сертификат. Обычной (но не обязательной) практикой при наличии разделяемого ключа является использование этого ключа для аутентификации в обоих направлениях. Отметим, что общепринято, хотя и не обеспечивает достаточной защиты, использование разделяемого ключа, созданного исключительно на базе выбранного пользователем пароля без использования других источников случайных данных.

Такая защита недостаточна, поскольку пользовательские пароли с очевидностью не являются достаточно непредсказуемыми для устойчивости к атакам по словарю, следовательно данный метод от таких атак не защищает (приложениям, использующим аутентификацию по паролю на этапе загрузки и IKE_SA, следует применять метод аутентификации, описанный в параграфе 2.16, который предназначен для защиты от атак по словарю в режиме off-line). Разделяемый (pre-shared) ключ **следует** делать столь же непредсказуемым, как наиболее сильный из согласуемых ключей. В случае pre-shared-ключа значение AUTH вычисляется, как:

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

где строка «Key Pad for IKEv2» представляет собой 17 символов ASCII без завершающего нуля. Разделяемый секрет (shared secret) может иметь переменный размер. Строка заполнения добавляется для того, чтобы при использовании разделяемого секрета на основе пароля реализации IKE не требовалось сохранять пароль в открытом виде, а можно было хранить его в форме prf(Shared Secret,"Key Pad for IKEv2"), которая не будет использоваться в качестве пароля для отличных от IKEv2 протоколов. Как отмечено выше, создание разделяемого ключа на основе пароля не обеспечивает должной защиты. Такая конструкция отмечена лишь потому, что многие люди ей пользуются до сих пор. Интерфейс управления, через который обеспечивается Shared Secret, **должен** принимать строки символов ASCII размером, по крайней мере, 64 октета; добавление завершающего нуля перед использованием строки в качестве разделяемого секрета **недопустимо**. Интерфейс управления **должен** также воспринимать разделяемый секрет в шестнадцатеричном (HEX) представлении. Интерфейс управления **может** воспринимать другие варианты кодировки строки, если указан алгоритм преобразования в двоичную строку. Если согласованная функция prf принимает ключ фиксированного размера, разделяемый секрет **должен** иметь такой же размер.

2.16. Методы EAP

В дополнение к аутентификации на основе подписей с открытыми ключами и разделяемыми секретами IKE поддерживает аутентификацию с использованием методов, определённых в RFC 3748 [EAP]. Обычно эти методы являются асимметричными (они разработаны для аутентификации пользователей на сервере) и могут не быть обоюдными. По это причине упомянутые протоколы обычно используются для аутентификации инициатора на отвечающей стороне и **должны** применяться в комбинации с аутентификацией ответчика инициатору по цифровой подписи на основе открытого ключа. Эти методы часто ассоциируются с механизмами, которые называют «унаследованной аутентификацией» (Legacy Authentication).

Хотя в этом документе [EAP] упоминается, прежде всего, в плане добавления в будущем новых методов без обновления данной спецификации, некоторые простые варианты описаны здесь и в параграфе 3.16. [EAP] определяет протокол аутентификации с переменным числом сообщений. Расширяемая аутентификация реализуется в IKE, как дополнительные обмены IKE_AUTH, которые **должны** быть выполнены для инициализации IKE_SA.

Инициатор показывает своё намерение использовать расширяемую аутентификацию, пропуская элемент AUTH в сообщении 3. За счёт включения элемента IDi при отсутствии AUTH инициатор объявляет свою идентификацию, но не подтверждает её. Если ответчик желает использовать расширяемую аутентификацию, он будет включать элемент EAP[1] в сообщение 4 и откладывает передачу SAr2, TSi и Tsr, пока аутентификации инициатора не будет завершена в последующем обмене IKE_AUTH. В варианте минимально расширяемой аутентификации организация начальной SA будет иметь вид:

```
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 разделяемый ключ в качестве побочного продукта аутентификации, этот ключ **должен** использоваться инициатором и ответчиком для генерации элементов данных AUTH в сообщениях 7 и 8 с использованием синтаксиса для разделяемых секретов, описанного в параграфе 2.15. Разделяемый ключ от EAP в спецификации EAP называется полем MSK. Разделяемый ключ, созданный в процессе обмена IKE, **недопустимо** использовать для иных целей.

Методы EAP, не создающие разделяемого ключа, использовать **не следует**, поскольку они подвержены многочисленным атакам MITM[2] [EAPMITM] при использовании в других протоколах, не применяющих аутентифицированные сервером туннели. Если используются методы EAP, не генерирующие разделяемого ключа, элементы данных AUTH в сообщениях 7 и 8 **должны** генерироваться с использованием SK_pi и SK_pr, соответственно.

[1] Extensible Authentication Protocol - протокол расширяемой аутентификации.
[2] Man-in-the-middle - атака с перехватом данных, в котором участвует человек.

Инициатору IKE_SA с использованием EAP **следует** поддерживать возможность расширения начального протокольного обмена по крайней мере до десяти обменов IKE_AUTH, если ответчик передаёт уведомления и/или повторяет приглашение к аутентификации. После успешного завершения протокольного обмена, определённого выбранным методом аутентификации EAP ответчик **должен** передать данные EAP, содержащие сообщение Success. Если при использовании выбранного метода аутентификации возник отказ, ответчик **должен** передать данные EAP, содержащие сообщение Failure. Ответчик может в любой момент прервать обмен IKE путём передачи данных EAP с сообщением Failure.

При таком расширенном обмене элементы данных EAP AUTH **должны** включаться в два сообщения, следующие за сообщением, содержащим EAP Success.

2.17. Материал для генерации ключей CHILD_SA

Одна связь CHILD_SA создаётся обменом IKE_AUTH, а дополнительные CHILD_SA могут создаваться в обменах CREATE_CHILD_SA. Ключевой материал для них создаётся следующим образом:

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

Здесь Ni и Nr - nonce из обмена IKE_SA_INIT, если данный запрос является первым созданием CHILD_SA, или обновлённые Ni и Nr из обмена CREATE_CHILD_SA при создании дополнительных связей.

Для обменов CREATE_CHILD_SA, включающих дополнительный обмен Diffie-Hellman, ключевой материал определяется следующим образом:

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

где g^ir (new) - разделяемый секрет из краткосрочного обмена Diffie-Hellman данного обмена CREATE_CHILD_SA (представляется, как строка октетов в формате big endian, дополненная нулями в старших битах при необходимости выравнивания размера по модулю).

Согласование одной CHILD_SA может приводить к созданию множества защищённых связей. Связи ESP и AH существуют попарно (по одной для каждого направления) и при использовании одновременно ESP и AH в одном согласовании CHILD_SA могут создаваться четыре SA.

Ключевой материал **должен** браться из KEYMAT в следующем порядке:

все ключи для SA, передающих данные от инициатора к ответчику, берутся до SA в обратном направлении;

если согласуется множество протоколов IPsec, ключевой материал для каждого из них берётся в порядке появления протокольных заголовков в инкапсулированном пакете;

если один протокол имеет ключи для шифрования и аутентификации, ключ шифрования берётся из первых октетов KEYMAT, а ключ аутентификации - из последующих.

Каждый криптографический алгоритм берет фиксированное число битов ключевого материала, задаваемое в спецификации алгоритма.

2.18. Смена ключей IKE_SA с использованием обмена CREATE_CHILD_SA

Обмен CREATE_CHILD_SA можно использовать для смены ключей существующей связи IKE_SA (см. 2.8). Новые SPI инициатора и ответчика передаются в полях SPI. Элементы данных TS опускаются при смене ключей IKE_SA. SKEYSEED для новой IKE_SA рассчитывается с использованием SK_d из существующей IKE_SA:

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

где g^ir (new) - разделяемый секрет из краткосрочного обмена Diffie-Hellman данного обмена CREATE_CHILD_SA (представляется, как строка октетов в формате big endian, дополненная нулями в старших битах при необходимости выравнивания размера по модулю), а Ni и Nr - два значения nonce из любых заголовков.

Новая связь IKE_SA **должна** сбросить свои счётчики сообщений в 0.

SK_d, SK_ai, SK_ar, SK_ei и SK_er рассчитываются из SKEYSEED, как описано в параграфе 2.14.

2.19. Запрос внутреннего адреса удалённой сети

В наиболее распространённом сценарии с подключением конечной точки к защитному шлюзу конечной точке может потребоваться адрес IP из защищённой шлюзом сети; для такого адреса может также потребоваться динамическое выделение. Запрос на такой временный адрес может быть включён в любой запрос на создание CHILD_SA (включая неявный запрос в сообщении 3) с помощью элемента данных CP.

Эта функция обеспечивает выделение адреса для клиента IRAC[1], пытающегося организовать туннель в сеть, защищённую сервером удалённого доступа IRAS[2]. Поскольку обмен IKE_AUTH создаёт связи IKE_SA и CHILD_SA, IRAC **должен** запрашивать контролируемый IRAS адрес (и, возможно, другую информацию о защищённой сети) в обмене IKE_AUTH. IRAS может предоставлять адрес для IRAC из любого числа источников адресов типа серверов DHCP/BOOTP или из собственного блока 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 элемент данных CP **должен** помещаться перед элементом SA. В вариациях протокола с множеством обменов IKE_AUTH элементы CP **должны** помещаться в сообщения, содержащие элементы SA.

[1] IPsec Remote Access Client - клиент удалённого доступа IPsec.
[2] IPsec Remote Access Server - сервер удалённого доступа IPsec.

Элемент CP(CFG_REQUEST) **должен** содержать по крайней мере атрибут INTERNAL_ADDRESS (IPv4 или IPv6), но может включать любое число дополнительных атрибутов, которые инициатор пожелал получить в отклике.

Ниже показан пример сообщения от инициатора к 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 TS содержат (протокол, диапазон портов, диапазон адресов).

Сообщение инициатору от ответчика:

```
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 возвращаемые значения зависят от реализации. Как можно видеть из приведённого выше примера, IRAS **может** также передавать другие атрибуты, которые не были включены в CP(CFG_REQUEST) и **могут** игнорировать необязательные атрибуты, которые они не поддерживают.

Для ответчика **недопустимо** передавать CFG_REPLY, если не был до этого принят запрос CP(CFG_REQUEST) от инициатора, поскольку мы не хотим, чтобы IRAS выполнял ненужные просмотры конфигурации, если IRAC не может обработать REPLY. В тех случаях, когда конфигурация IRAS требует, использования CP для данного IDi, но IRAC не удалось передать CP(CFG_REQUEST), сервер IRAS **должен** отвергнуть запрос и прервать обмен IKE с возвратом ошибки FAILED_CP_REQUIRED.

2.20. Запрос версии партнёра

Узел IKE, желающий узнать номер версии программ своего партнёра IKE, **может** воспользоваться описанным ниже методом. Это пример конфигурационного запроса в рамках обмена INFORMATIONAL после создания IKE_SA и первой дочерней CHILD_SA.

Реализация IKE **может** отвергать запросы на выдачу своего номера версии до завершения аутентификации партнёра и даже после аутентификации в целях предотвращения возможности использования известных недостатков в части защиты. В таких случаях реализация **должна** возвращать пустую строку или пакет без 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

При обработке IKE может происходить множество разных ошибок. Если принятый запрос имеет некорректный формат или неприемлем по соображениям политики (например, не соответствуют криптографические алгоритмы), отклик **должен** содержать элемент Notify, показывающий ошибку. Если ошибка произошла за пределами контекста запроса IKE (например, узел получает сообщения ESP на несуществующий SPI), узлу **следует** инициировать обмен INFORMATIONAL с элементом данных Notify, описывающим проблему.

Ошибки, которые происходят до организации криптографически защищённой IKE_SA, должны обрабатываться с особой осторожностью. Существует компромисс между желанием оказать пользу в диагностике и решении проблем и желанием избежать возможности стать жертвой атаки на отказ служб в результате реакции на обманные сообщения.

Если узел получает сообщение в порт UDP с номером 500 или 4500 за пределами известного ему контекста IKE_SA (и это сообщение не является запросом на создание контекста), это может говорить о недавней аварии узла. Если сообщение отмечено, как отклик, узел **может** занести информацию о нем в журнал аудита, но отвечать на такое сообщение **недопустимо**. Если сообщение помечено, как запрос, отклик на него **должен** быть передан в адрес IP и порт, с которых запрос поступил, при этом значения IKE SPI и Message ID в отклике должны быть копиями этих полей из запроса. **Недопустимо** использовать для отклика криптографическую защиту и отклик **должен** содержать элемент Notify, показывающий INVALID_IKE_SPI.

Узлу, получившему такой незащищённый элемент Notify, **недопустимо** менять состояние существующих SA. Сообщение может быть обманным или может являться легитимного корреспондента, вовлечённого в передачу обманным путём. Узлу **следует** трактовать такое сообщение (а также сетевые сообщения типа ICMP destination unreachable), как намёк о возможности проблем с SA для данного адреса IP, а также **следует** выполнить проверку жизненности для всех таких IKE_SA. Реализациям **следует** ограничивать частоту таких проверок для предотвращения возможности их использования для организации атак на службы.

Узел, получивший подозрительное сообщение с IP-адреса, с которым он имеет IKE_SA, **может** передать элемент данных IKE Notify в обмене IKE INFORMATIONAL через имеющуюся SA. Получателю **недопустимо** менять состояние каких-либо SA в результате приёма такого сообщения, но **следует** записать событие в журнал аудита для упрощения диагностики. Узел **должен** ограничивать скорость передачи откликов на незащищённые сообщения.

2.22. Компрессия IPComp

Использование компрессии IP [IPCOMP] может быть согласовано на этапе создания CHILD_SA. Хотя компрессия IP включает дополнительный заголовок в каждом пакете и список параметров компрессии (CPI1), виртуальная «связь с

компрессией» не существует за пределами содержащей её ESP или AH SA. «Связи с компрессией» исчезают при удалении соответствующей ESP или AH SA. Эти связи не упоминаются явно в элементах данных DELETE.

Согласование компрессии IP отделено от согласования криптографических параметров, связанных с CHILD_SA. Узел, запрашивающий создание CHILD_SA, **может** анонсировать поддержку одного или множества алгоритмов компрессии путём передачи одного или множества элементов Notify типа IPCOMP_SUPPORTED. Отклик **может** показывать приемлемость одного алгоритма компрессии с помощью элемента Notify типа IPCOMP_SUPPORTED. Такие элементы **недопустимо** включать в сообщения, не содержащие элементов SA.

Хотя выше говорилось о допустимости использования множества алгоритмов компрессии и возможности использовать разные алгоритмы для двух направления CHILD_SA, данная спецификация **запрещает** реализациям принимать алгоритм IPComp, который не был предложен, а также **запрещает** использовать алгоритмы, кроме того, который был предложен и принят на этапе создания CHILD_SA.

Побочным эффектом раздельного согласования IPComp и криптографических параметров является невозможность предложить множество криптографических наборов и компрессию IP, которая будет использоваться с частью (но не со всеми) предложенных наборов.

2.23. Работа через NAT

Использование шлюзов с трансляцией сетевых адресов (NAT) является спорным вопросом. В этом параграфе кратко описаны такие шлюзы и их действия по отношению к трафику IKE. Многие враждебно относятся к NAT и считают, что не следует разрабатывать протоколы для улучшения работы через системы трансляции адресов. IKEv2 задаёт некоторые не вполне очевидные правила обработки для улучшения работы через NAT.

Основной причиной использования NAT является нехватка адресов IPv4. Узлы IP, находящиеся за шлюзами NAT используют адреса IP, которые не являются уникальными в глобальном масштабе и могут совпадать с адресами, которые применяются за другими шлюзами NAT. В общем случае узлы, расположенные за NAT, могут взаимодействовать с другими узлами за тем же шлюзом NAT и узлами с уникальными в глобальном масштабе адресами, не не с узлами, расположенными за другим шлюзом NAT. Из этого правила есть ряд исключений. Когда узлы, расположенные за NAT, соединяются с узлами в реальной сети Internet, шлюз NAT «преобразует» (транслирует) IP-адрес отправителя, меняя его на адрес, который будет маршрутизироваться обратно на этот шлюз. Полученные из Internet сообщения «транслируются» шлюзом с заменой адреса получателя на внутренний адрес соответствующего конечного узла.

Система NAT разрабатывалась с учётом обеспечения прозрачности для оконечных узлов. Ни программы находящегося за NAT узла, ни узлы Internet не требуется менять для работы через NAT. Обеспечение такой прозрачности для одних протоколов сложнее, чем для других. Протоколы, включающие IP-адреса конечных точек в данные (не только в заголовки), будут сталкиваться с проблемами, пока шлюз NAT не начнёт понимать протокол и соответствующим образом изменять данные в пакетах. Такое решение является изначально ненадёжным, нарушает целостность сетевого уровня и часто приводит к возникновению трудно обнаруживаемых проблем.

Организация соединений IPsec через NAT вызывает определённые проблемы. Если соединение работает в транспортном режиме, изменение адресов IP в пакетах будет приводить к изменению контрольных сумм, которые NAT не сможет скорректировать, поскольку они криптографически защищены. Даже в туннельном режиме возникают проблемы с маршрутизацией, поскольку прозрачная трансляция адресов в пакетах AH и ESP требует реализации в NAT специальной логики, которая по своей природе эвристична и ненадёжна. По этим причинам IKEv2 использует инкапсуляцию пакетов IKE и ESP в UDP. Такой вариант слегка снижает эффективность, но проще для обработки в NAT. Кроме того, межсетевые экраны могут быть настроены на передачу трафика IPsec, инкапсулированного в UDP, но при этом блокировать ESP/AH и наоборот.

NAT обычно используется для трансляции портов TCP и UDP, наряду с адресами, и использования номеров портов из входящих пакетов для принятия решения об адресе локального узла, которому адресован пакет. По этой причине, хотя пакеты IKE **должны** отправляться через порт UDP с номером 500, **необходимо** принимать пакеты, исходящие из любого порта и отклики **должны** направляться в порт, из которого был получен запрос. Эти требования обусловлены тем, что номера портов могут меняться при прохождении пакетов через NAT. Аналогично, адреса IP конечных точек IKE обычно не включаются в элементы данных IKE, поскольку данные криптографически защищены и не могут прозрачно изменяться устройствами NAT.

Порт 4500 зарезервирован для инкапсуляции ESP и IKE в UDP. При работе через NAT в общем случае лучше передавать пакеты IKE через порт 4500, поскольку некоторые старые реализации NAT обрабатывают трафик IKE через порт 500 некорректно, пытаясь организовать прозрачное соединение IPsec между между конечными точками, которые сами по себе не поддерживают работу через NAT. Такие реализации NAT могут конфликтовать с прямым прохождением через NAT, описанным в этом документе, поэтому конечная точка IPsec, которая обнаружит NAT между собой и своим корреспондентом, **должна** передавать весь последующий трафик через порт 4500, который NAT не следует обрабатывать специальным способом (как это может происходить для порта 500).

Ниже перечислены специфические требования для прохождения через NAT [RFC3715]. Поддержка работы через NAT является необязательной. Приведённые в этом параграфе требования типа **«должно»** относятся только к реализациям, поддерживающим работу через NAT.

IKE **должен** прослушивать порты 4500 и 500. IKE **должен** отвечать по адресам IP и портам, с которых были приняты пакеты.

Как инициатор, так и ответчик IKE **должны** включать в свои пакеты IKE_SA_INIT элементы данных Notify типа NAT_DETECTION_SOURCE_IP и NAT_DETECTION_DESTINATION_IP. Эти элементы могут использоваться для детектирования наличия NAT между хостами и определения, какой из хостов находится за NAT. Эти элементы располагаются в пакетах IKE_SA_INIT после элементов Ni и Nr (перед необязательным элементом CERTREQ).

Если ни один из полученных элементов NAT_DETECTION_SOURCE_IP не соответствует хэшу IP-адреса и порта отправителя в заголовке IP содержащего элемент пакета, это означает, что другая сторона расположена за NAT

(где-то на пути доставки адрес отправителя исходного пакета заменён на адрес устройства NAT). В таких случаях данной стороне следует разрешить динамическое обновление IP-адреса другой стороны, как описано ниже.

Если полученный элемент данных NAT_DETECTION_DESTINATION_IP не соответствует хэшу IP-адреса и номера порта получателя из заголовка IP пакета, содержащего элемент данных, это означает, что другая сторона расположена за NAT. В этом случае локальной стороне **следует** начать передачу пакетов keepalive, как описано в [Hutt05].

Инициатор IKE **должен** проверить наличие этих элементов данных и при несоответствии адресам во внешнем пакете **должен** туннелировать все будущие пакеты IKE и ESP, связанные с данной IKE_SA через порт UDP 4500.

При туннелировании пакетов IKE через порт UDP 4500 заголовок IKE имеет четыре октета нулей, следующих сразу после заголовка UDP. При туннелировании пакетов ESP через порт UDP 4500 заголовок ESP следует сразу после заголовка UDP. Поскольку первые четыре байта заголовка ESP содержат значение SPI, которое не может быть нулевым, это позволяет всегда легко различать сообщения ESP и IKE.

Исходные IP-адреса отправителя и получателя, которые нужны в транспортном режиме для расчёта контрольной суммы пакетов TCP и UDP (см. [Hutt05]), извлекаются из селекторов трафика, связанных с этим обменом. При работе через NAT селекторы трафика **должны** содержать в точности один адрес IP, который используется в качестве исходного адреса IP.

В некоторых случаях устройство NAT может удалить отображения, которые продолжают использоваться (например, интервал keepalive слишком велик или устройство NAT перезагружено). Для восстановления в таких случаях хостам, которые не расположены за NAT, **следует** передавать все пакеты (включая повторные) в адрес IP и порт из последнего корректно аутентифицированного пакета от другой стороны (т. е., динамически обновлять адрес). Расположенному за NAT хосту **не следует** делать так, поскольку это будет открывать возможность организации DoS-атак. Любой аутентифицированный пакет IKE или ESP, инкапсулированный в UDP, можно использовать для детектирования смены IP-адреса и номера порта. Отметим, что похожие, но, возможно, не совсем идентичные, действия требуется выполнять при работе IKE через Mobile IP, но этот вопрос выходит за пределы данного документа.

2.24. Явные уведомления о перегрузке (ECN)

При развёртывании туннелей IPsec в соответствии с исходной спецификацией [RFC2401], использование ECN во внешних заголовках IP было, по сути, невозможно, поскольку при декапсуляции туннеля индикаторы перегрузки ECN, появившиеся в сети, отбрасывались. Поддержка ECN для туннелей IPsec на базе IKEv1 требует множества режимов работы и согласований (см. [RFC3168]). IKEv2 упрощает ситуацию, вводя требование возможности использования ECN во внешних заголовках IP для всех IPsec SA в туннельном режиме, создаваемых IKEv2. В частности, точки инкапсуляции и декапсуляции туннельных SA, создаваемых IKEv2, **должны** поддерживать опцию полной функциональности ECN для туннелей, заданную в [RFC3168], а также **должны** реализовать инкапсуляцию и декапсуляцию в соответствии с [RFC4301] для предотвращения отбрасывания индикации перегрузки ECN.

3. Форматы заголовков и данных

3.1. Заголовок IKE

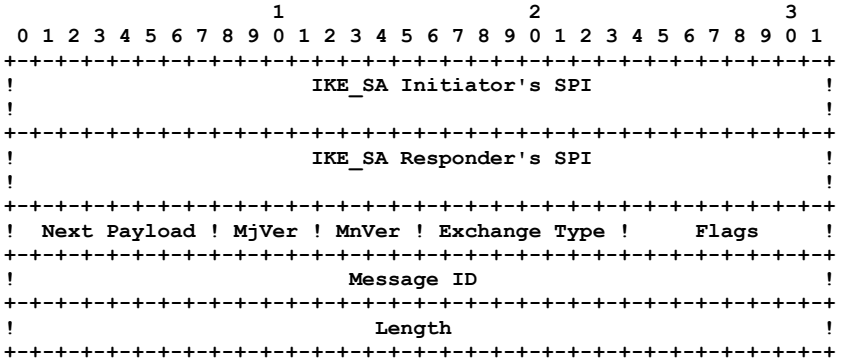

Рисунок 4. Формат заголовка IKE.

Сообщения IKE используют протокол UDP через порты 500 и/или 4500, передаётся по одному сообщению IKE в дейтаграмме UDP. Информация от начала пакета до завершения заголовка UDP большей частью игнорируется, исключения составляют адреса IP и номера портов UDP в заголовках, которые сохраняются и используются для передачи ответных пакетов. При передаче через порт UDP 500 сообщение IKE начинается непосредственно после заголовка UDP. При передаче через порт UDP 4500 перед сообщением IKE помещается четыре октета с нулевыми значениями. Эти октеты не являются частью сообщения IKE и не учитываются в размерах и контрольных суммах IKE. Каждое сообщение IKE начинается с заголовка IKE, обозначаемого в данном документе HDR. После заголовка следует один или множество элементов данных IKE, каждый из которых идентифицируется полем Next Payload в предыдущем элементе данных. Элементы данных обрабатываются в порядке их следования в сообщении IKE путём вызова соответствующей процедуры, согласно значению поля Next Payload в заголовке IKE, потом согласно значению Next Payload в первом элементе данных IKE и так далее, пока в поле Next Payload не будет обнаружено нулевое значение, показывающее отсутствие следующего элемента данных. При обнаружении элемента данных типа Encrypted этот элемент дешифруется и результат расшифровки разбирается, как дополнительные элементы данных. Элемент Encrypted **должен** быть последним элементом в пакете и включать в зашифрованные элементы другие элементы типа Encrypted **недопустимо**.

Значение Recipient SPI в заголовке идентифицирует экземпляр защищённой связи IKE. Следовательно, один экземпляр IKE может мультиплексировать различные сессии с множеством партнёров.

Многооктетные поля, представляющие собой целые числа используют сетевой порядок байтов (или big endian - старший байт сначала).

Рисунок 4 показывает формат заголовка IKE.

- **Initiator's SPI** (8 октетов) - значение, выбранное инициатором для уникальной аутентификации защищённой связи IKE. Нулевое значение **недопустимо**.
- **Responder's SPI** (8 октетов) - значение, выбранное ответчиком для уникальной аутентификации защищённой связи IKE. Это значение **должно** быть нулевым в первом сообщении начального обмена IKE (включая повторы этого сообщения, содержащие cookie), для всех остальных сообщений нулевое значение **недопустимо**.
- **Next Payload** (1 октет) - показывает тип элемента данных, расположенного сразу после заголовка. Форматы и значения всех типов описаны ниже.
- **Major Version** (4 бита) - задаёт старшую часть номера версии используемого протокола IKE. Реализации на основе данной версии IKE, **должны** устанавливать Major Version=2. Реализации, основанные на предыдущих версиях IKE и ISAKMP, **должны** устанавливать Major Version=1. Основанные на этой версии протокола IKE реализации **должны** отвергать или игнорировать пакеты со значением этого поля, превышающим 2.
- **Minor Version** (4 бита) - задаёт младшую часть номера версии IKE. Реализации на основе этого документа **должны** устанавливать Minor Version = 0 и игнорировать младшую часть номера в принимаемых сообщениях.
- **Exchange Type** (1 октет) - показывает тип обмена, который будет использоваться. Тип ограничивает набор элементов данных в каждом сообщении и порядок сообщений в обмене. Типы показаны в таблице.

Тип обмена	*Значение*
Резерв	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
Резерв IANA	38-239
Резерв для частного использования	240-255

- **Flags** (1 октет) - показывает специфические опции, установленные для сообщения. Наличие опции указывается установкой соответствующего флага. Биты флагов начинаются с младшего, т. е. Бит 0 является младшим битом октета Flags. В приведённом ниже описании термин «установлен» означает значение бита 1, а термин «сброшен» - значение 0.
  - **X**(резерв) (биты 0-2) - эти биты **должны** сбрасываться при передаче и игнорироваться на приёме.
  - **I**(nitiator) (бит 3 поля Flags) - этот бит **должен** устанавливаться в сообщениях, передаваемых исходным инициатором IKE_SA, и **должен** сбрасываться в сообщениях, передаваемых исходным ответчиком. Этот бит позволяет получателю определить, какие восемь октетов SPI были созданы получателем.
  - **V**(ersion) (бит 4 поля Flags) - этот флаг показывает, что передающий узел способен поддерживать большее значение старшей части номера версии, нежели указано в поле Major Version. Реализации IKEv2 **должны** сбрасывать этот бит при передаче и игнорировать его во входящих сообщениях[1].
  - **R**(esponse) (бит 5 поля Flags) - этот бит показывает, что данное сообщение является откликом на сообщение с таким же идентификатором. Этот бит **должен** сбрасываться во всех запросах и устанавливаться во всех откликах. Для конечной точки IKE **недопустима** генерация откликов на сообщения, помеченные, как отклик.
  - **X**(резерв) (биты 6-7) - эти биты **должны** сбрасываться при передаче и игнорироваться на приёме.
- **Message ID** (4 октета) - идентификатор сообщения служит для управления повторной передачей потерянных пакетов и связывания запросов с откликами. Это поле важно для безопасности протокола, поскольку оно используется для предотвращения атак с повторным использованием перехваченных пакетов (replay-атаки). См. также параграфы 2.1 и 2.2.
- **Length** (4 октета) - размер всего сообщения (заголовок и элементы данных) в октетах.

3.2. Базовый заголовок элемента данных

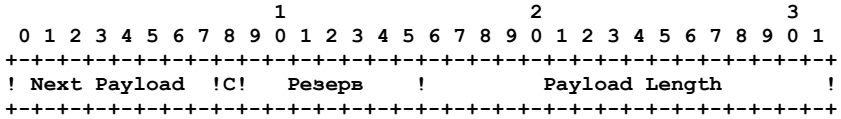

Рисунок 5. Формат базового заголовка данных.

Каждый из элементов данных (payload) IKE, определённых в параграфах 3.3 - 3.16, начинается с базового заголовка[2] (Рисунок 5). Рисунки в описании каждого элемента включают базовый заголовок, но описания полей этого заголовка для краткости опущены и приводятся только в этом параграфе.

- **Next Payload** (1 октет) - идентификатор типа следующего элемента данных в сообщении. Если текущий элемент является последним, это поле имеет значение 0. Это поле позволяет создавать «цепочки», когда дополнительный элемент просто добавляется в конец сообщения и устанавливается значение поля Next Payload в предыдущем элементе для индикации типа нового элемента. Элемент типа Encrypted, который всегда должен быть последним в сообщении, является исключением. Он содержит структуры данных в

[1] В оригинале это предложение содержит ошибку, см. https://www.rfc-editor.org/errata_search.php?eid=1671. *Прим. перев.*
[2] Generic Payload Header.

формате дополнительных элементов. В заголовке элемента Encrypted поле Next Payload устанавливается в соответствии с типом первого вложенного элемента (вместо 0).

Значения Payload Type показаны в таблице.

Тип Next Payload	*Обозначение*	*Значение*
No Next Payload		0
Резерв		1-32
Security Association	SA	33
Key Exchange	KE	34
Identification - Initiator	IDi	35
Identification - Responder	IDr	36
Certificate	CERT	37
Certificate Request	CERTREQ	38
Authentication	AUTH	39
Nonce	Ni, Nr	40
Notify	N	41
Delete	D	42
Vendor ID	V	43
Traffic Selector - Initiator	TSi	44
Traffic Selector - Responder	TSr	45
Encrypted	E	46
Configuration	CP	47
Extensible Authentication	EAP	48
Резерв IANA		49-127
Для частного применения		128-255

Значения типов 1-32 не следует использовать во избежание перекрытия со значениями, применяемыми в IKEv1. Типы 49-127 зарезервированы IANA для будущего распределения в IKEv2 (см. параграф 6). Типы 128-255 выделены для частного применения по согласованию сторон.

- **Critical** (1 бит) - это поле **должно** иметь значение 0, если отправитель хочет, чтобы получатель пропустил этот элемент, если он не понимает код в поле Next Payload предыдущего элемента. Если получатель понимает тип элемента, он **должен** игнорировать этот флаг. Это поле **должно** устанавливаться в 0 для определённых в документе типов элементов данных. Отметим, что флаг критичности относится к текущему элементу данных, а не к следующему, чей тип указывается в первом октете. Причина сбрасывания бита критичности для определённых здесь элементов заключается в том, что все реализации **должны** поддерживать все типы элементов, определённые в этой спецификации, и, следовательно, должны игнорировать значение флага Critical. Предполагается, что пропускаемые элементы будут иметь корректные значения полей Next Payload и Payload Length.
- **RESERVED** (7 битов) - **должно** иметь нулевое значение при передаче и игнорироваться на приёме.
- **Payload Length** (2 октета) - размер текущего элемента данных в октетах с учётом базового заголовка.

3.3. Элементы данных SA

Данные защищённой связи[1], обозначаемые в этом документе SA, служат для согласования атрибутов защищённой связи. Сборка элементов данных SA требует внимания. Элемент SA **может** включать множество предложений. Если предложений больше одного, они **должны** быть упорядочены в порядке снижения предпочтительности. Каждое предложение может включать множество протоколов IPsec (протоколами являются IKE, ESP, AH), каждый протокол **может** включать множество преобразований, а каждое преобразование **может** включать множество атрибутов. При разборе SA реализация **должна** проверить соответствие значения Payload Length размерам и числу отдельных компонент. Предложения (Proposal), преобразования (Transform) и атрибуты (Attribute) используют своё представление с различными размерами. Они вкладываются в элемент так, чтобы значение поля Payload Length элемента SA учитывало данные SA, Proposal, Transform и Attribute. Размер Proposal включает размер всех содержащихся в нем Transform и Attribute. Размер Transform включает размеры всех содержащихся в нем Attribute.

Синтаксис элементов SA, Proposal, Transform и Attribute основан на ISAKMP, однако семантика их слегка отличается. Причина использования иерархической структуры заключается в том, что такая структура позволяет представлять в одной SA множество возможных комбинаций алгоритмов. Иногда предоставляется выбор из множества алгоритмов, в других случаях - комбинация алгоритмов. Например, инициатор может предложить использование комбинации (AH w/MD5 **И** ESP w/3DES) **ИЛИ** (ESP w/MD5 **И** 3DES).

Одной из причин изменения семантики элементов SA по сравнению с ISAKMP и IKEv1 является повышение уровня компактности представления в наиболее распространённых случаях.

Структура Proposal включает Proposal # (номер предложения) и идентификатор протокола IPsec. Каждая структура **должна** использовать значение Proposal #, совпадающее со значением в предыдущей структуре или увеличенное на 1. Первый элемент Proposal **должен** иметь Proposal # = 1. Если две структуры подряд имеют одинаковый номер предложения, это значит, что предложение включает первую **И** вторую структуры[2]. Так, для предложения AH **И** ESP будут использоваться две структуры (одна для AH, другая для ESP) с одинаковым номером Proposal #1. Для предложения AH **ИЛИ** ESP будут использоваться две структуры - для AH с номером Proposal #1 и для ESP с номером Proposal #2.

За каждой структурой Proposal/Protocol следует одна или множество структур преобразований. Число разных преобразований обычно определяется элементом Protocol. Протокол AH обычно имеет одно преобразование - алгоритм контроля целостности. ESP обычно имеет два преобразования - алгоритм шифрования и алгоритм контроля целостности. IKE в общем случае имеет четыре преобразования - группа Diffie-Hellman, алгоритм контроля целостности, алгоритм prf и алгоритм шифрования. Если предлагаются комбинированные алгоритмы шифрования и

[1] Security Association Payload.
[2] Пересечение или операция **И** (AND). *Прим. перев.*

защиты целостности, он **должен** предлагаться, как алгоритм шифрования, а предлагать в таком случае алгоритм защиты целостности **недопустимо**. Для каждого протокола набору допустимых преобразования присваиваются идентификаторы, включаемые в заголовок каждого преобразования.

Если имеется множество преобразований одного типа (одно значение Transform Type), они должны комбинироваться в одно предложение с помощью операции **ИЛИ**[1]. При наличии множества преобразований различных типов, группы объединяются операцией **И**. Например, чтобы предложить ESP с (3DES или IDEA) и (HMAC_MD5 или HMAC_SHA), предложение ESP будет включать два кандидата Transform Type 1 (один для 3DES, второй для IDEA) и два кандидата Transform Type 3[2] (для HMAC_MD5 и HMAC_SHA). Это обеспечивает эффективное предложение четырёх комбинаций алгоритмов. Если инициатор хочет предложить только подмножество этого - например, (3DES и HMAC_MD5) или (IDEA и HMAC_SHA), - не существует возможности представить это в виде множества преобразований в одном предложении. Взамен инициатор будет создавать два разных предложения по паре преобразований в каждом.

Преобразование **может** иметь один или множество атрибутов (Attribute). Атрибуты требуются, когда преобразование может использоваться множеством способов (например, алгоритм шифрования с переменным размером ключей - в этом случае преобразование будет задавать алгоритм, а атрибут - размер ключа). Большинство преобразований не имеет атрибутов. Для преобразования **недопустимо** наличие множества однотипных атрибутов. Чтобы предложить варианты значения для атрибута (например, множество размеров ключа для алгоритма шифрования AES), реализация **должна** включать множество преобразований с общим значением Transform Type, каждое из которых имеет один атрибут (Attribute).

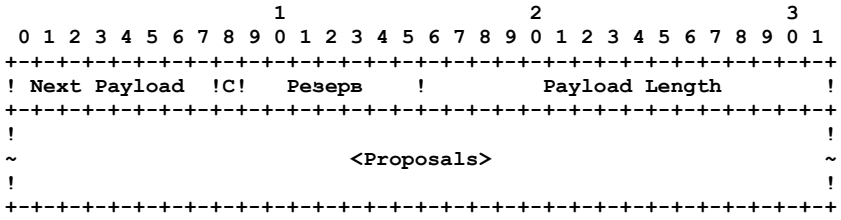

Рисунок 6. Элемент данных SA.

Отметим, что семантика Transform и Attribute достаточно сильно отличается от IKEv1. В IKEv1 одно преобразование задаёт множество алгоритмов для протокола и один из них передаётся в Transform, а другие в Attribute.

- **Proposals** (переменный размер) - одна или множество субструктур Proposal.

  Идентификатор типа для элемента SA имеет значение 33.

3.3.1. Субструктура Proposal

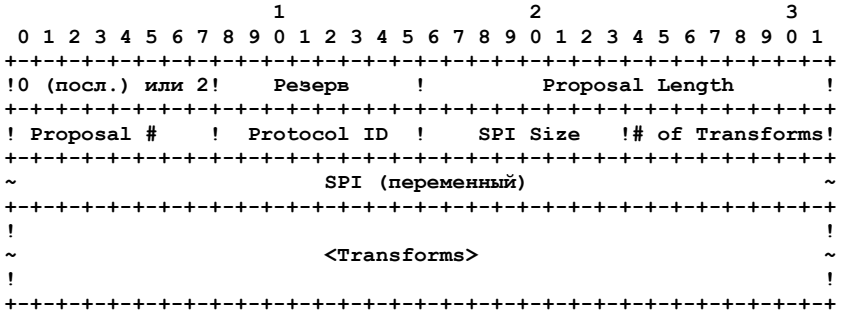

Рисунок 7. Субструктура Proposal.

- **0 (последнее) или 2 (не последнее) (1 октет)** - показывает, является ли предложение последним в субструктуре предложений элемента SA. Синтаксис унаследован от ISAKMP, не это поле не является необходимым, поскольку последнее предложение можно идентифицировать по размеру SA. Значение 2 соответствует типу элемента Proposal в IKEv1 и первые четыре октета структуры Proposal организованы так, чтобы они выглядели, подобно заголовку Payload.
- **Резерв** (1 октет) - **должно** устанавливаться в 0 при передаче и игнорироваться на приёме.
- **Proposal Length** (2 октета) - размер данного предложения, включая все входящие в него преобразования и атрибуты.
- **Proposal #** (1 октет) - номер предложения. Первое предложение в элементе SA **должно** иметь номер 1, а номера последующих должны совпадать с номером предшественника (**И** - пересечение двух предложений) или быть на 1 больше (**ИЛИ** - объединение двух предложений). Когда предложение принимается, все номера предложений в элементе SA **должны** совпадать и соответствовать номеру переданного предложения, которое было принято.
- **Protocol ID** (1 октет) - задаёт идентификатор протокола IPsec для текущего согласования. Определённые значения идентификаторов показаны в таблице.

Протокол	*Protocol ID*
Резерв	0
IKE	1
AH	2
ESP	4

[1]В https://www.rfc-editor.org/errata_search.php?eid=2192 была отмечена логическая ошибка в этом предложении, однако текст был сохранен и в последующих версиях документа - RFC 5996 и RFC 7296. *Прим. перев.*

[2]В оригинале ошибочно указано Transform Type 2 (см. типы преобразований ниже). В в последующих версиях документа - RFC 5996 и RFC 7296 - тип преобразования был указан корректно. *Прим. перев.*

Резерв IANA	4 - 200
Частное применение	201 - 255

- **SPI Size** (1 октет) - для начального согласования IKE_SA это поле **должно** иметь нулевое значение; значение SPI получается из внешнего заголовка. При последующих согласованиях это поле показывает размер (в октетах) SPI для соответствующего протокола (8 для IKE, 4 для ESP и AH).
- **# of Transforms** (1 октет) - показывает число преобразований в данном предложении.
- **SPI** (переменный размер) - SPI передающей стороны. Даже если значение SPI Size не кратно 4, для элементов данных не используется заполнения. При нулевом значении SPI Size это поле не включается в элемент SA.
- **Transforms** (переменный размер) - одна или множество субструктур Transform.

3.3.2. Субструктура Transform

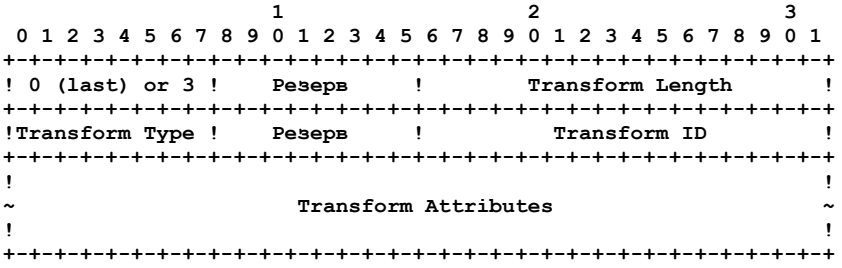

Рисунок 8. Субструктура Transform.

- 0 (последнее) или 3 (не последнее) (1 октет) - показывает, является ли это преобразование последним в предложении. Синтаксис унаследован от ISAKMP, не это поле не является необходимым, поскольку последнее предложение можно идентифицировать по размеру Proposal[1]. Значение 2 соответствует типу элемента Transform в IKEv1 и первые четыре октета структуры организованы так, чтобы они выглядели, подобно заголовку Payload.
- **Резерв** (1 октет) - **должно** устанавливаться в 0 при передаче и игнорироваться на приёме.
- **Transform Length** - размер субструктуры Transform (в октетах) с учётом заголовка и атрибутов.
- **Transform Type** (1 октет) - показывает тип преобразования, задаваемого этим элементом. Различные протоколы поддерживают разные типы преобразований. Для некоторых протоколов часть преобразований может быть опциональной. Если преобразование является необязательным и инициатор предлагает его пропустить, преобразования этого типа не включаются в предложение. Если инициатор желает отдать решение вопроса об использовании необязательного преобразования ответчику, он включает субструктуру этого преобразования с нулевым идентификатором (Transform ID = 0) в качестве одной из опций.
- **Transform ID** (2 октета) - указывает конкретный экземпляр предлагаемого преобразования.

Типы преобразований перечислены ниже в таблице.

	Тип преобразования	***Используется***
Резерв	0	
Encryption Algorithm (ENCR) - алгоритм шифрования	1	IKE и ESP
Pseudo-random Function (PRF) - псевдослучайная функция	2	IKE
Integrity Algorithm (INTEG) - алгоритм защиты целостности	3	IKE, AH, опционально в ESP
Diffie-Hellman Group (D-H) - группа Diffie-Hellman	4	IKE, опционально в AH и ESP
Extended Sequence Numbers (ESN) - расширенные порядковые номера	5	AH и ESP
Резерв IANA	6 - 240	
Частное применение	241-255	

Для преобразований типа 1 (Transform Type 1 - алгоритм шифрования) определены показанные в таблице идентификаторы (Transform ID).

Имя	***Значение***	***Определение***
Резерв	0	
ENCR_DES_IV64	1	RFC1827
ENCR_DES	2	RFC2405, [DES]
ENCR_3DES	3	RFC2451
ENCR_RC5	4	RFC2451
ENCR_IDEA	5	RFC2451, [IDEA]
ENCR_CAST	6	RFC2451
ENCR_BLOWFISH	7	RFC2451
ENCR_3IDEA	8	RFC2451
ENCR_DES_IV32	9	
Резерв	10	
ENCR_NULL	11	RFC2410
ENCR_AES_CBC	12	RFC3602
ENCR_AES_CTR	13	RFC3664
Резерв IANA	14 - 1023	
Резерв для частного использования по соглашению сторон	1024-65535	

Для преобразований типа 2 (псевдослучайная функция) определены идентификаторы, показанные в таблице.

[1] В оригинале это предложение содержит ошибку, см. https://www.rfc-editor.org/errata_search.php?eid=1672. *Прим. перев.*

Имя	*Значение*	*Определение*
Резерв	0	
PRF_HMAC_MD5	1	RFC2104, [MD5]
PRF_HMAC_SHA1	2	RFC2104, [SHA]
PRF_HMAC_TIGER	3	RFC2104
PRF_AES128_XCBC	4	RFC3664
Резерв IANA	5 - 1023	
Резерв для частного использования по соглашению сторон	1024-65535	

Для преобразований типа 3 (алгоритм защиты целостности) определены идентификаторы, показанные в таблице.

Имя	*Значение*	*Определение*
NONE	0	
AUTH_HMAC_MD5_96	1	RFC2403
AUTH_HMAC_SHA1_96	2	RFC2404
AUTH_DES_MAC	3	
AUTH_KPDK_MD5	4	RFC1828[1]
AUTH_AES_XCBC_96	5	RFC3566
Резерв IANA	5 - 1023	
Резерв для частного использования по соглашению сторон	1024-65535	

Для преобразований типа 4 (группа Diffie-Hellman) определены идентификаторы, показанные в таблице.

Имя	*Значение*
NONE	0
Определены в Приложении B	1 - 2
Резерв	3 - 4
Определены в [ADDGROUP]	5
Резерв IANA	6 - 13
Определены в [ADDGROUP]	14 - 18
Резерв IANA	19 - 1023
Частное применение	1024-65535

Для преобразований типа 5 (расширенные порядковые номера) определены идентификаторы, показанные в таблице.

Имя	*Значение*
No Extended Sequence Numbers - нет расширенных номеров	0
Extended Sequence Numbers - расширенные номера	1
Резерв	2 - 65535

3.3.3. Приемлемые типы преобразований по протоколам

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

Число и тип преобразований в элементах SA зависит от типа протокола в самой SA. Элемент данных SA, предлагающий организацию SA имеет обязательные и опциональные типы преобразований. Совместимая с требованиями реализация **должна** понимать все обязательные и дополнительные типы для каждого поддерживаемого ею протокола (хотя принимать предложения с неподходящими наборами не требуется). Предложение **может** опускать необязательные типы, если единственным воспринимаемым значением этого типа является NONE.

3.3.4. Обязательные Transform ID

Спецификация наборов, которые **должно** и **следует** поддерживать в целях взаимодействия, была удалена из этого документа, поскольку стало очевидно, что эти наборы меняются быстрее, чем спецификация самого протокола.

Важным результатом использования IKEv1 является понимание того, что системам не следует реализовать только обязательные алгоритмы и ждать, что они явятся лучшим выбором для всех пользователей. Например, во время подготовки этого документа многие разработчики IKEv1 начали переход на AES в режиме CBC[2] для приложений VPN[3]. Многие системы IPsec на базе IKEv2 будут поддерживать AES, дополнительные группы Diffie-Hellman и дополнительные алгоритмы хэширования, а некоторым пользователям IPsec уже требуются эти алгоритмы в дополнение к перечисленным выше.

Очевидно, что IANA будет добавлять новые преобразования, а некоторые пользователи могут применять приватные шифронаборы, особенно для IKE, где разработчикам следует обеспечивать поддержку различных параметров, вплоть до некоторых ограничений размера. В поддержку этой идеи всем реализациям IKEv2 **следует** включать средства управления, которые позволяют (пользователю или системному администратору) задавать параметры Diffie-Hellman (генератор, модуль, размер и значения экспоненты) для новых групп DH. Разработчикам **следует** поддерживать интерфейс управления, через который могут задаваться эти параметры и связанные с ними Transform ID (пользователем или системным администратором) для обеспечения возможности согласования таких групп.

Все реализации IKEv2 **должны** включать средства управления, которые позволят пользователю или администратору системы задавать наборы, подходящие для использования с IKE. При получении элемента данных с набором Transform ID реализация **должна** сравнить переданные идентификаторы преобразований с выбранными локально для проверки согласованности предложенного набора с локальной политикой. Реализация **должна** отвергать предложения SA, которые не разрешены этими средствами управления наборами IKE. Отметим, что шифронаборы, которые **должны** быть реализованы, не требуется указывать в локальной политике, как приемлемые.

3.3.5. Атрибуты преобразования

[1] В оригинале ошибочно указан RFC 1826, см. https://www.rfc-editor.org/errata_search.php?eid=2279. *Прим. перев.*
[2] Cipher Block Chaining - цепочки шифрованных блоков.
[3] Virtual Private Network - виртуальная частная сеть.

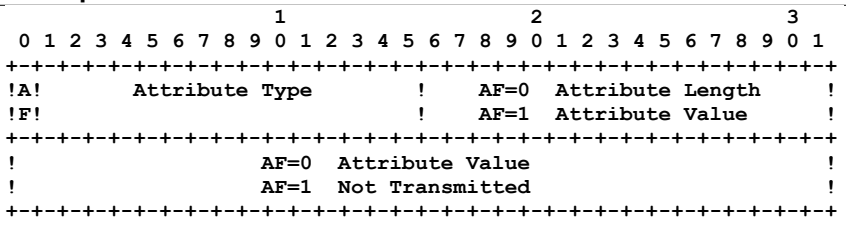

Рисунок 9. Атрибуты преобразования.

Каждое преобразование в элементе SA может включать атрибуты, меняющие или дополняющие спецификацию преобразования. Эти атрибуты представляют собой пары «тип-значение» и определены ниже. Например, если алгоритм шифрования имеет ключи переменного размера, этот размер может задаваться в качестве атрибута. Атрибуты могут иметь значение фиксированного (2 октета) или переменного размера. В последнем случае для представления атрибута используется формат «тип-размер-значение».

- **Attribute Type** (2 октета) - уникальный идентификатор для каждого типа атрибутов (см. ниже).

  Старший бит этого поля является флагом формата атрибута (AF[1]), показывающим использование полного (TLV[2]) или сокращённого (TV[3]) формата. При AF = 0, для атрибута используется формат TLV, при AF = 1 - TV.

- **Attribute Length** (2 октета) - размер поля Attribute Value в октетах. При AF = 1, размер Attribute Value всегда составляет 2 октета и поле Attribute Length отсутствует.

- **Attribute Value** (переменный размер) - значение атрибута, связанное с Attribute Type. Если AF = 0, размер этого поля указывается в поле Attribute Length. При AF = 1 размер поля Attribute Value всегда равен 2 октетам.

Отметим, что в настоящее время определён только один тип атрибута - размер ключа (Key Length) и для него используется фиксированный размер. Спецификации атрибутов переменной длины включены только для будущих расширений. Из определённых в этом документе алгоритмов атрибуты принимают только основанные на AES функции шифрования, защиты целостности и генерации случайных чисел - им нужен один атрибут, задающий размер ключа.

Атрибуты, описанные в качестве базовых, **недопустимо** представлять с использованием переменного размера. Атрибуты переменного размера **недопустимо** представлять в качестве базовых, даже если их значение может быть помещено в два октета. Это отличается от IKEv1 в том, что повышается гибкость и упрощается создание сообщений, но несколько усложняется их разбор.

Тип	*Значение*	*Формат*
Резерв	0-13	
Key Length (в битах)	14	TV
Резерв	15-17	
Резерв IANA	18-16383	
Для частного применения	16384-32767	

Значения 0-13 и 15-17 использовались в аналогичном контексте IKEv1 и их не следует выделять во избежание конфликтов. Значения 18-16383 зарезервированы для IANA. Диапазон 16384-32767 выделен для частного применения по согласованию сторон.

Key Length - размер ключа

При использовании алгоритма шифрования с ключами переменного размера этот атрибут показывает размер ключа в битах (**должен** использоваться сетевой порядок байтов). Этот атрибут **недопустимо** использовать с алгоритмами шифрования, имеющими фиксированный размер ключа.

3.3.6. Согласование атрибутов

При согласовании защищённой связи инициаторы вносит свои предложения ответчикам. Ответчики **должны** выбрать из предложений один полный набор параметров (или отвергнуть все предложения, если они не подходят). Если имеется множество предложений, ответчик **должен** выбрать номер одного из них и вернуть инициатору все субструктуры Proposal с этим номером предложения. Если имеется множество однотипных преобразований, ответчик **должен** выбрать одно из них. Все атрибуты выбранного преобразования **должны** возвращаться в неизменном виде. Инициатор обмена **должен** убедиться в том, что принятое предложение согласуется со сделанными им предложениями. При наличии расхождений отклик **должен** быть отвергнут.

Согласование групп Diffie-Hellman имеет некоторые особенности. SA включает предлагаемые атрибуты и открытое значение Diffie-Hellman (KE[4]) в одном сообщении. Если в начальном обмене инициатор предлагает использовать одну из нескольких групп Diffie-Hellman, ему **следует** выбрать ту, которую ответчик с высокой вероятностью примет, и включить соответствующее этой группе значение KE. Если предсказание окажется неверным, ответчик укажет в отклике корректную группу и инициатору **следует** найти для своего KE элемент в этой группе при повторе сообщения. Однако ему **следует** по-прежнему предлагать полный набор поддерживаемых групп, чтобы предотвратить возможность организации атак, направленных на снижение уровня защиты.

Примечание для разработчиков. Некоторые согласуемые атрибуты могут включать диапазоны, со множеством подходящих значений (например, размеры ключа переменной длины для симметричного шифра). Для повышения уровня взаимодействия и обеспечения возможности независимого обновления конечных точек разработчикам реализаций этого протокола следует принимать значения, которые они считают обеспечивающими лучшую защиту. Например, если партнёр настроен на принятие шифра с переменным размером ключа для размера X (битов) и предлагается более длинный ключ, реализации **следует** принять такое предложение, если она может работать с более длинным ключом.

[1]Attribute Format.
[2]Type/Length/Value - тип/размер/значение.
[3]Type/Value - тип/значение.
[4]Элемент данных Key Exchange. *Прим. перев.*

Поддержка такой возможности позволяет реализации выражать концепции защиты не ниже определённого уровня - «ключ размером **не менее** X битов для шифра Y».

3.4. Обмен ключами

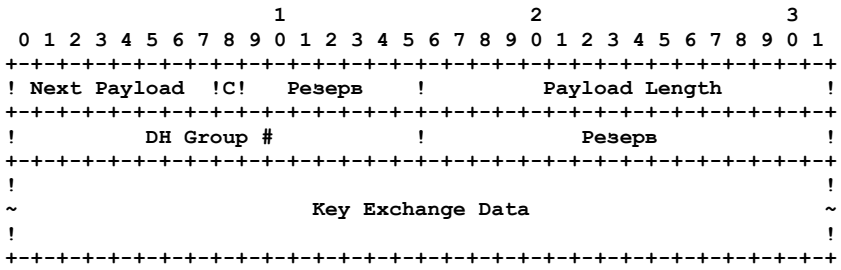

Рисунок 10. Формат элемента Key Exchange.

Элемент данных Key Exchange (KE) служит для обмена открытыми номерами Diffie-Hellman при обмене ключами Diffie-Hellman. Элемент KE включает базовый заголовок IKE, за которым следует открытое значение Diffie-Hellman.

Элемент данных обмена ключами создаётся путём копирования открытого значения Diffie-Hellman в поле Key Exchange Data. Размер открытого значения Diffie-Hellman **должен** быть равен размеру первичного модуля, для которого выполняется возведение в степень, дополненного при необходимости нулями в начале.

Поле DH Group # идентифицирует группу Diffie-Hellman в которой было рассчитано значение Key Exchange Data (см. параграф 3.3.2). Если выбранное предложение использует другую группу Diffie-Hellman, сообщение **должно** быть отвергнуто с возвратом элемента Notify типа INVALID_KE_PAYLOAD.

Тип элемента для обмена ключами KE имеет значение 34.

3.5. Идентификация

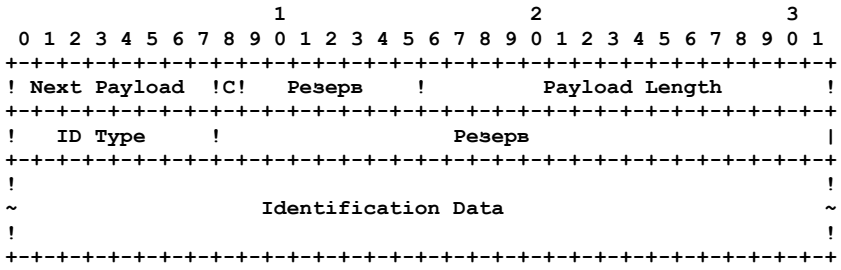

Рисунок 11. Формат идентификации.

Элемент данных Identification (ID), обозначаемый в этом документе IDi и IDr, позволяет партнёрам предъявлять свою идентификацию другой стороне. Эта идентификация может использоваться при просмотре политики, но не обязана соответствовать информации в элементе CERT - оба эти поля могут использоваться реализацией для контроля доступа.

Примечание. В IKEv1 использовались два элемента ID в каждом направлении для данных селекторов трафика (TS) для данных, передаваемого через SA. В IKEv2 эта информация передаётся в элементах TS (см. параграф 3.13).

- **ID Type** (1 октет) - задаёт тип используемой идентификации.
- **Резерв** - **должно** обнуляться при передаче и игнорироваться на приёме.
- **Identification Data** (переменный размер) - значение, указанное полем Identification Type. Размер данных идентификации рассчитывается по размеру в заголовке элемента ID.

Тип элемента для Identification может принимать значение 35 (Idi) и 36 (IDr).

В таблице приведён список выделенных значений поля ID Type с описанием соответствующего поля Identification Data.

ID Type	*Значение*	*Описание Identification Data*
Резерв	0	
ID_IPV4_ADDR	1	Один четырехоктетный адрес IPv4.
ID_FQDN	2	Строка полного доменного имени (например, example.com). В строку **недопустимо** включать символы завершения (NULL, CR и т. п.).
ID_RFC822_ADDR	3	Строка полного почтового адреса RFC822 (например, jsmith@example.com). В строку **недопустимо** включать символы завершения.
Резерв IANA	4	
ID_IPV6_ADDR	5	Один шестнадцатиоктетный адрес IPv6.
Резерв IANA	6-8	
ID_DER_ASN1_DN	9	Двоичное представление в формате DER[1] для ASN.1 X.500 Distinguished Name [X.501].
ID_DER_ASN1_GN	10	Двоичное представление в формате DER для ASN.1 X.500 GeneralName [X.509].
ID_KEY_ID	11	Неструктурированный поток октетов, который может использоваться для передачи связанной с производителем информации, требуемой для некоторых фирменных вариантов идентификации.
Резерв IANA	12-200	

[1]Distinguished Encoding Rules.

Резерв для частного использования	201-255	

Две реализации будут совместимы только в том случае, когда каждая может генерировать тип ID, приемлемый для другой стороны. Для обеспечения максимальной совместимости реализация **должна** быть настраиваемой на передачу по крайней мере одного из типов ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR, ID_KEY_ID и восприятие всех этих типов. Реализациям **следует** обеспечивать возможность генерации и восприятия всех этих типов. Поддерживающие IPv6 реализации **должны** дополнительно иметь возможность настройки восприятия ID_IPV6_ADDR. Реализации, поддерживающие только IPv6, **можно** настраивать на передачу только ID_IPV6_ADDR.

3.6. Сертификат

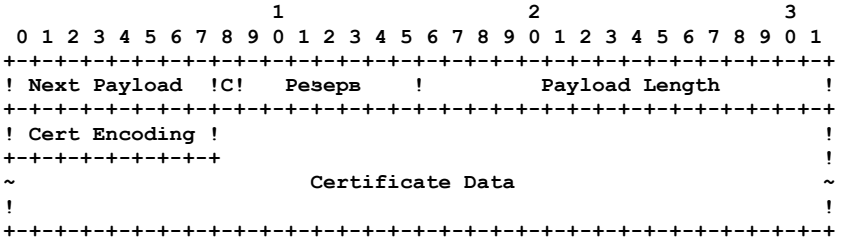

Рисунок 12. Формат сертификата.

Элемент данных Certificate (CERT) обеспечивает способ передачи сертификатов или других, связанных с аутентификацией данных через IKE. Элементы Certificate **следует** включать в обмен, если сертификаты доступны отправителю до того, как партнёр указал возможность получения аутентификационной информации иным путём с использованием элемента Notify типа HTTP_CERT_LOOKUP_SUPPORTED. Отметим, что термин «Certificate Payload» может вводить в заблуждение, поскольку не все механизмы аутентификации используют сертификаты и в этом элементе могут передаваться иные данные.

Кодирование сертификата	***Значение***
Резерв	0
Сертификат X.509 с PKCS #7	1
Сертификат PGP	2
Подписанный ключ DNS	3
Сертификат X.509 - подпись	4
Маркер Kerberos	6
CRL	7
ARL	8
Сертификат SPKI	9
Сертификат X.509 - атрибут	10
Неразобранный ключ RSA	11
Хэш и URL сертификата X.509	12
Хэш и URL связки (bundle) X.509	13
Резерв IANA	14 - 200
Для частного применения	201 - 255

Элемент данных Certificate имеет следующие поля:

- **Certificate Encoding** (1 октет) - это поле показывает тип представления сертификата или иной информации, содержащейся в поле Certificate Data (см. таблицу на врезке справа).
- **Certificate Data** (переменный размер) - представление данных сертификата. Тип сертификата указывается в поле Certificate Encoding.

Идентификатор типа элемента данных Certificate имеет значение 37.

Конкретный синтаксис ряда перечисленных типов сертификатов в этом документе не определяется. К числу сертификатов, синтаксис которых определён здесь относятся:

Сертификат X.509 - подпись (4) содержит сертификат X.509 (в представлении DER), открытый ключ которого используется для проверки элемента данных AUTH отправителя.

Список отозванных сертификатов (7) содержит представление DER для списка отозванных сертификатов X.509.

Неразобранный ключ RSA (11) содержит ключ RSA в представлении PKCS #1 (см. [RSA] и [PKCS1]).

Хэш и URL[1] (12-13) позволяют включать в сообщения IKE замену больших структур данных с 20-октетным хэшем SHA-1 (см.[SHA]) значением URL переменной длины, которое преобразуется в структуру данных (в представлении DER). Это повышает эффективность в тех случаях, когда конечные точки имеют хэшированные сертификаты и снижает эффект воздействия на IKE атак на отказ служб, которые становились бы проще в реализации при использовании достаточно больших сообщений IKE, требующих фрагментации на уровне IP [KPS03].

Ниже приведено представление сборки X.509 в формате ASN.1.

```
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
```

[1]Uniform Resource Locator – однотипный указатель ресурсов. *Прим. перев.*

```
  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
```

Реализации **должны** обеспечивать возможность настройки передачи и восприятия до четырёх сертификатов X.509 в поддержку аутентификации, а также **должны** обеспечивать возможность настройки передачи и восприятия двух первых форматов Hash and URL (с HTTP URL). Реализациям **следует** обеспечивать возможность настройки передачи и восприятия ключей Raw RSA. При передаче множества сертификатов первый из них **должен** содержать открытый ключ, используемый для подписывания элемента AUTH. Остальные сертификаты можно передавать в любом порядке.

3.7. Запрос сертификата

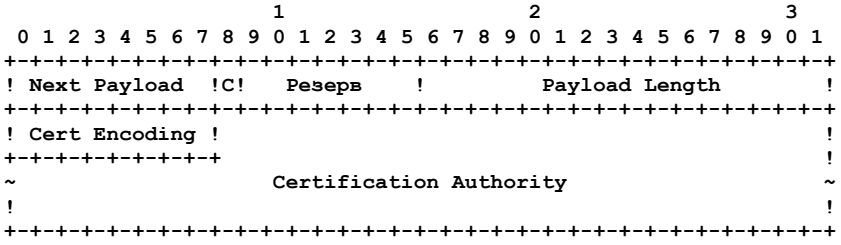

Рисунок 13. Формат запроса сертификата.

Элемент Certificate Request (CERTREQ) обеспечивает возможность запросить предпочитаемые сертификаты через IKE и может появляться в откликах IKE_INIT_SA и запросах IKE_AUTH. Элементы Certificate Request **могут** включаться в обмен, когда отправителю нужен сертификат получателя. При наличии множества доверенных CA[1], если поле Cert Encoding не разрешает список, **следует** передавать множество элементов Certificate Request.

Элемент Certificate Request содержит следующие поля:

- **Certificate Encoding** (1 октет) - задаёт тип представления запрашиваемого сертификата. Возможные значения перечислены в параграфе 3.6.
- **Certification Authority** (переменный размер) - указывает подходящий удостоверяющий центр для запрашиваемого типа сертификата.

Идентификатор типа для элемента данных Certificate Request имеет значение 38.

Поле Certificate Encoding имеет такие же значения, как описано в параграфе 3.6. Поле Certification Authority содержит индикатор удостоверяющего центра для данного типа сертификата. Значение Certification Authority представляет собой конкатенацию хэш-значений SHA-1 для открытых ключей удостоверяющих центров (CA). Каждое значение представляет собой хэш SHA-1 для элемента Subject Public Key Info (см. параграф 4.1.2.7 работы [RFC3280]) из каждого сертификата Trust Anchor. Двадцатиоктетные хэш-значения объединяются (конкатенация) и помещаются в поле без дополнительного форматирования.

Отметим, что термин Certificate Request (запрос сертификата) может вводить в заблуждение, поскольку запрашиваться с помощью этого элемента могут не только сертификаты, но и другие данные, как было указано в описании элемента Certificate. Синтаксис элемента Certificate Request для таких случаев не определяется в этом документе.

Элемент Certificate Request обрабатывается путём проверки поля Cert Encoding для определения наличия у обрабатывающего сертификатов этого типа. Если такие сертификаты имеются, просматривается поле Certification Authority для проверки наличия у обрабатывающего сертификатов, которые могут быть подтверждены в одном из указанных удостоверяющих центров. Это может быть цепочка сертификатов.

Если существует сертификат конечного объекта, который удовлетворяет критериям, заданным в CERTREQ, этот сертификат или цепочку сертификатов **следует** передать назад запрашивающему сертификат узлу, при условии, что получатель CERTREQ:

- настроен на использование аутентификации по сертификатам;
- имеет разрешение на передачу элемента CERT;
- имеет соответствующую политику доверия к CA для текущего согласования;
- имеет по крайней мере один действующий (time-wise) и подходящий сертификат конечного объекта, связанный с CA из CERTREQ.

Проверка сертификатов на предмет их отзыва должна выполняться в процессе создания цепочек, используемом для выбора сертификата. Отметим, что даже при настройке двух партнёров. на использование разных CA, в логике выбора следует поддерживать отношения кросс-сертификации.

Не ставится задачи блокирования связи на основе строгого соответствия выбора сертификата предложенному в CERTREQ - отправитель может выбирать другие сертификаты, которые получатель может проверить и которым он сможет доверят на основе кросс-сертификации, списков отзыва и других способов. Таким образом, обработке CERTREQ следует выглядеть, как предложению на выбор сертификата (не обязательно одного). Если сертификата нет, элемент CERTREQ игнорируется. С точки зрения протокола это не является ошибкой. Могут возникать случаи,

[1]Certification Authority - удостоверяющий центр. *Прим. перев.*

когда при наличии предпочтительного CA, указанного в CERTREQ, подходящим оказывается другой удостоверяющий центр (возможно, в результате выбора оператора).

3.8. Аутентификация

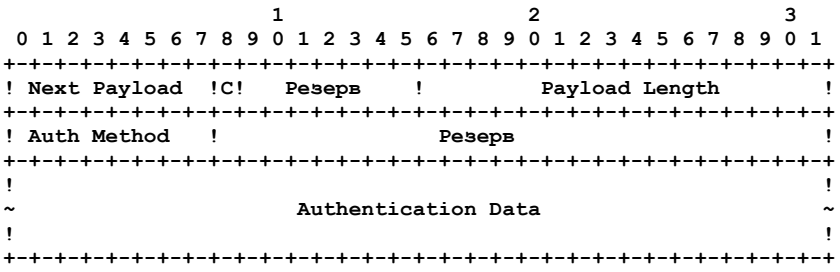

Рисунок 14. Формат аутентификации.

Элемент данных Authentication (AUTH) содержит данные, которые используются для аутентификации. Синтаксис Authentication Data меняется в соответствии с выбором Auth Method, как описано ниже.

Элемент Authentication имеет следующие поля:

- **Auth Method** (1 октет) - задаёт используемый метод аутентификации и может принимать значения:

  RSA Digital Signature (1) - рассчитывается, как указано в параграфе 2.15, с использованием приватного ключа RSA и хэш-значения PKCS#1 с заполнением (см. [RSA] и [PKCS1]).

  Shared Key Message Integrity Code (2) - рассчитывается, как указано в параграфе 2.15, с использованием разделяемого ключа, связанного с объектом из элемента ID, и согласованной функции prf.

  DSS Digital Signature (3) - рассчитывается, как указано в параграфе 2.15, с использованием приватного ключа DSS (см. [DSS]) и хэш-значения SHA-1.

  Значение 0 и 4-200 зарезервированы IANA. Значения 201-255 выделены для частных приложений.

- **Authentication Data** (переменный размер) - см. параграф 2.15.

Идентификатор типа элемента Authentication имеет значение 39.

3.9. Элемент Nonce

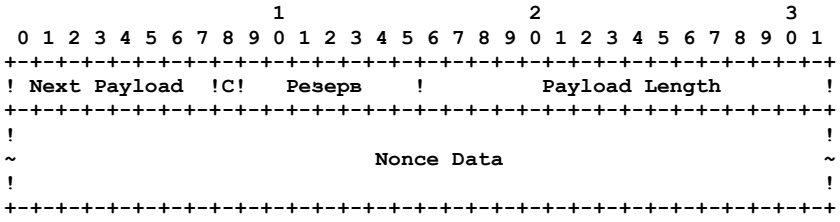

Рисунок 15. Формат Nonce.

Элементы Nonce, обозначаемые в этом документе Ni и Nr (для инициатора и ответчика, соответственно), содержат случайные значения, служащие для обеспечения жизнестойкости в процессе обмена и защиты от атак с повторным использованием пакетов.

Элемент Nonce имеет одно поле:

- **Nonce Data** (переменный размер) - случайное значение, созданное передающей стороной.

Идентификатор типа элемента Nonce имеет значение 40.

Размер Nonce **должен** находиться в диапазоне от 16 до 256 октетов, включительно. **Недопустимо** повторное использование значений Nonce.

3.10. Уведомление

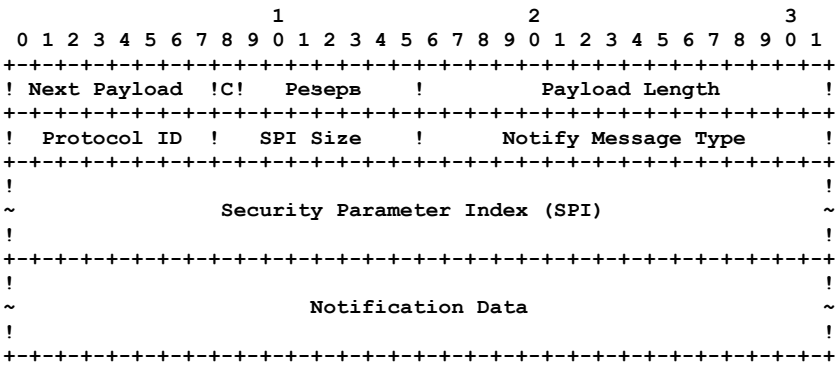

Рисунок 16. Формат уведомления.

Элемент Notify (N) используется для передачи служебной информации (ошибки, смена состояния) партнёру. IKE. Элемент Notify может появляться в откликах на отвергнутые запросы (обычно с указанием причины отказа), обмене

INFORMATIONAL (для сообщений об ошибках, не относящихся к запросу IKE) и в других сообщениях для индикации возможностей отправителя или изменения трактовки запроса.

Элемент Notify имеет следующие поля:

- **Protocol ID** (1 октет) - если уведомление относится к существующей SA, это поле указывает тип данной SA. Для уведомлений IKE_SA это поле **должно** иметь значение 1. Для уведомлений, касающихся IPsec SA, это поле **должно** содержать значение 2 для протокола AH или 3 для ESP. Для уведомлений, не относящихся к существующим SA, это поле **должно** сбрасываться в 0 при передаче и игнорироваться на приёме. Все остальные значения зарезервированы IANA для использования в будущем.
- **SPI Size** (1 октет) - размер SPI (в октетах) в соответствии с идентификатором протокола IPsec или 0, если SPI не применим. Для уведомлений, касающихся IKE_SA, поле SPI Size **должно** иметь значение 0.
- **Notify Message Type** (2 октета) - задаёт тип уведомления.
- **SPI** (переменный размер) - список параметров защиты.
- **Notification Data** (переменный размер) - информация или данные об ошибке, передаваемые в дополнение к Notify Message Type. Значения этого поля зависят от типа и описаны ниже.

Идентификатор типа элемента Notify имеет значение 41.

3.10.1. Типы уведомлений

Уведомления могут сообщать о причинах ошибок, не позволивших организовать SA. Они могут также содержать данные о состоянии, которые процесс, управляющий базой данных SA, желает передать процессу партнёра. Приведённая ниже таблица содержит список сообщений Notification и их идентификаторов. Число различных ошибочных состояний существенно снижено по сравнению с IKEv1 в целях упрощения и сокращения объёма информации для зондирования[1].

Типы 0 - 16383 предназначены для передачи сообщений об ошибках. Реализация, получившая элемент Notify с одним из таких типов, который она не способна распознать, при отклике **должна** предполагать, что соответствующий запрос не удалось обработать совсем. Непонятные типы ошибок в запросах и типы состояний в запросах и откликах **должны** игнорироваться, но при этом их **следует** заносить в системный журнал.

Элементы Notify с типами, относящимися к состоянию, **можно** добавлять в любые сообщения. Непонятные элементы таких типов **должны** игнорироваться. Такие сообщения предназначены для индикации возможностей и, как часть процесса согласования SA, используются для согласования параметров, не являющихся криптографическими.

Сообщение Notify - ошибки	*Значение*	*Описание*
Резерв	0	
UNSUPPORTED_CRITICAL_PAYLOAD	1	Передаётся в тех случаях, когда не распознан элемент с флагом Critical. Поле Notification Data содержит октет типа элемента.
INVALID_IKE_SPI	4	Показывает, сообщение IKE, полученное с нераспознанным SPI адресата. Это обычно говорит о перезагрузке партнёра с утратой существующей IKE_SA.
INVALID_MAJOR_VERSION	5	Показывает, что получатель не может обрабатывать версию IKE, указанную в заголовке. В заголовке отклика указывается ближайший номер версии, поддерживаемой получателем.
INVALID_SYNTAX	7	Указывает сообщения полученные IKE, которые неприемлемы по причине выхода за допустимые пределы типа, размера или значения, а также были отвергнуты по соображениям политики. Для предотвращения атак на службы с использованием обманных сообщений этот код можно возвращать только для шифрованных сообщений (в шифрованных сообщениях), если идентификатор сообщения и криптографическая контрольная сумма корректны. Во избежание утечки информации к зондирующему это сообщение **должно** передаваться в ответ на любую ошибку, для которой не выделено кода. Для отладки **следует** выводить более детальную информацию об ошибке на консоль или в системный журнал.
INVALID_MESSAGE_ID	9	Передаётся при получении сообщения IKE, идентификатор которого не попадает в поддерживаемое окно. Такое уведомление **недопустимо** передавать в отклике, поскольку подтверждение некорректного запроса **недопустимо**. Вместо этого другую сторону можно проинформировать с помощью обмена INFORMATIONAL с поле Notification, содержащим четыре октета некорректного идентификатора сообщения. Передача этого уведомления является опциональной и **должна** быть ограничена по частоте.
INVALID_SPI	11	**Может** передаваться в обмене INFORMATIONAL, когда узел получает пакет ESP или AH с некорректным SPI. Поле Notification Data содержит SPI из полученного пакета. Обычно такое сообщение говорит о перезагрузке узла с потерей SA. Если такое сообщение передаётся вне контекста IKE_SA, его следует использовать только в качестве «совета», поскольку подделать такое сообщение достаточно просто.
NO_PROPOSAL_CHOSEN	14	Ни один из предложенных шифронаборов не может быть принят.

[1]Например, с целью организации атак. *Прим. перев.*

INVALID_KE_PAYLOAD	17	Поле D-H Group # элемента KE не совпадает с номером группы, выбранным ответчиком для этого обмена. С таким уведомлением связаны два октета (в сетевом порядке) данных, указывающие номер приемлемой группы D-H.
AUTHENTICATION_FAILED	24	Передаётся в ответ на сообщение IKE_AUTH, когда аутентификация не прошла. Связанных данных нет.
SINGLE_PAIR_REQUIRED	34	Это сообщение показывает, что запрос CREATE_CHILD_SA не принят потому, что отправитель согласен принимать только селекторы трафика, задающие одну пару адресов. Предполагается, что запрашивающая сторона ответит запросом SA только для конкретного трафика.
NO_ADDITIONAL_SAS	35	Это сообщение показывает, что запрос CREATE_CHILD_SA не принят потому, что ответчик не хочет создавать дополнительные CHILD_SA для данной IKE_SA. Некоторые минимальные реализации могут принимать организацию только одной CHILD_SA в контексте начального обмена IKE и отвергают все последующие попытки организации связей.
INTERNAL_ADDRESS_FAILURE	36	Показывает ошибку при выделении внутреннего адреса (INTERNAL_IP4_ADDRESS или INTERNAL_IP6_ADDRESS) в процессе обработки ответчиком элемента Configuration. Если это сообщение создано в контексте обмена IKE_AUTH, связи CHILD_SA не будут организованы.
FAILED_CP_REQUIRED	37	Передаётся ответчиком в тех случаях, когда CP(CFG_REQUEST) предполагалось, но не было получено в результате конфликта с локальной политикой. Связанных данных нет.
TS_UNACCEPTABLE	38	Показывает неприемлемость всех адресов/протоколов/портов в селекторе трафика.
INVALID_SELECTORS	39	**Может** передаваться в обмене INFORMATIONAL, когда узел получает пакет ESP или AH, в котором селекторы не соответствуют использованной SA (это приводит к отбрасыванию пакета). Поле Notification Data содержит начало ошибочного пакета (как в сообщениях ICMP), а в поле SPI помещается значение SPI для IPsec SA.
Резерв IANA - типы ошибок	40 - 8191	
Для частного применения - типы ошибок	8192 - 16383	

Сообщения NOTIFY - состояния	***Значение***	***Описание***
INITIAL_CONTACT	16384	Данное уведомление заявляет, что данная IKE_SA является единственной активной IKE_SA между аутентифицированными сторонами. Уведомление **может** передаваться при создании IKE_SA после аварии и получатель **может** использовать эту информацию для удаления всех прочих IKE_SA с тем же аутентифицированным партнёром без ожидания. **Недопустима** передача таких уведомлений со стороны реплицируемых объектов (например, представление мобильного пользователя, которому разрешено подключаться к корпоративному шлюзу с двух удалённых систем одновременно).
SET_WINDOW_SIZE	16385	Это уведомление заявляет, что передающая сторона способна сохранять состояние для множества незавершённых обменов, что позволяет получателю отправлять множество запросов без ожидания ответа на первый запрос. Данные, связанные с SET_WINDOW_SIZE, **должны** иметь размер 4 октета (сетевой порядок байтов) и содержать представление числа сообщений, которые могут сохраняться. До завершения начального обмена размер окна всегда равен 1.
ADDITIONAL_TS_POSSIBLE	16386	Это уведомление заявляет, что передающая сторона сужает предложенный набор селекторов трафика, и говорит о возможности использования других селекторов через отдельные SA (см. параграф 2.9). С этим типом уведомлений не связано данных. Уведомление может передаваться только в качестве дополнительного элемента сообщения, включающего подходящие TS.

IPCOMP_SUPPORTED	16387	Это уведомление можно включать только в сообщения, содержащие элемент SA для согласования CHILD_SA, где указано желание использовать IPComp в данной SA. Связанные с уведомлением данные включают двухоктетное значение IPComp CPI, за которым следует одноктетный идентификатор преобразования (возможно, с атрибутами, размер и формат которых определяются идентификатором преобразования). Сообщение, предлагающее SA, может включать множество уведомлений IPCOMP_SUPPORTED для индикации поддержки разных алгоритмов. Сообщение, принимающее SA, может содержать не более одного такого уведомления. Идентификаторы преобразований показаны ниже. **Имя** **Номер** **Определение** Резерв 0 IPCOMP_OUI 1 IPCOMP_DEFLATE 2 RFC 2394 IPCOMP_LZS 3 RFC 2395 IPCOMP_LZJH 4 RFC 3051 Значения 5-240 зарезервированы IANA, значения 241-255 предназначены для часто применения по согласования сторон.
NAT_DETECTION_SOURCE_IP	16388	Это уведомление используется получателем для проверки нахождения его отправителя за устройством NAT. Данные, связанные с этим уведомлением, представляют собой подпись SHA-1 для значений SPI (в порядке их следования в заголовке), адреса IP и номера порта, куда пакет был послан. **Возможно** наличие множества уведомлений этого типа в сообщении, если отправитель не знает, какое из соединений с сетью будет использоваться для передачи пакета. Получатель уведомления **может** сравнить полученное значение с хэшем SPI, адреса IP и порта получателя - при несовпадении **следует** включить режим работы через NAT (см. параграф 2.23). Если работа через NAT не поддерживается, попытка соединения **может** быть отвергнута.

Сообщения NOTIFY - состояния	***Значение***	***Описание***
NAT_DETECTION_DESTINATION_IP	16389	Это уведомление используется его получателем для проверки своего нахождения за устройством NAT. Данные, связанные с этим уведомлением, представляют собой подпись SHA-1 для значений SPI (в порядке их следования в заголовке), адреса IP и номера порта, куда пакет был послан. Получатель уведомления **может** сравнить полученное значение с хэшем SPI, адреса IP и порта получателя - при несовпадении **следует** включить режим работы через NAT (см. параграф 2.23). Несоответствие говорит о том, что данная сторона расположена за устройством NAT и ей **следует** начать передачу пакетов keepalive, как определено в [Hutt05]. Если работа через NAT не поддерживается, попытка соединения **может** быть отвергнута.
USE_TRANSPORT_MODE	16391	Это уведомление **может** быть включено в запрос, содержащий также элемент SA для создания CHILD_SA. Уведомление запрашивает для CHILD_SA использование транспортного режима, вместо туннельного[1]. Если запрос принимается, отклик на него **должен** включать уведомление USE_TRANSPORT_MODE. Если ответчик отвергает запрос, CHILD_SA будет создаваться для туннельного режима. Если это неприемлемо для инициатора, он **должен** удалить SA. Примечание. Для всех SA с туннельным режимом, создаваемых IKEv2, **должна** выполняться декапсуляция, изменённая в соответствии с [RFC4301].
HTTP_CERT_LOOKUP_SUPPORTED	16392	Это уведомление **может** включаться в любое сообщение, которое содержит элемент CERTREQ, и показывает, что отправитель может находить сертификаты по URL на основе HTTP (и, следовательно, будет предпочитать получение сертификатов в таком формате).
REKEY_SA	16393	Это уведомление **должно** включаться в обмен CREATE_CHILD_SA, если задачей обмена является замена существующей SA (ESP или AH). Поле SPI аутентифицирует SA, для которой будут меняться ключи. Уведомление не включает данных/
ESP_TFC_PADDING_NOT_SUPPORTED	16394	Это уведомление заявляет, что передающая точка **не** будет принимать пакеты с заполнением TFC[2].
NON_FIRST_FRAGMENTS_ALSO	16395	Служит для контроля фрагментации (см. [RFC4301]).
Резерв IANA - типы состояний	16396 - 40959	
Для частного применения - типы состояний	40960 - 65535	

[1]Без использования такого уведомления все CHILD_SA создаются для работы в туннельном режиме.
[2]Flow Confidentiality - конфиденциальность потока трафика. *Прим. перев.*

3.11. Удаление

Элемент Delete (D) содержит определяемый протоколом идентификатор защищённой связи, которую отправитель удаляет из базы данных (следовательно, связь перестаёт быть корректной). Рисунок 17 Показывает формат элемента Delete. В этом элементе данных можно передавать множество SPI, однако все SPI **должны** относиться к одному протоколу. Смешивание идентификаторов протоколов в элементе Delete **недопустимо**. Однако в одном обмене INFORMATIONAL допускается использование множества уведомлений Delete с SPI для разных протоколов.

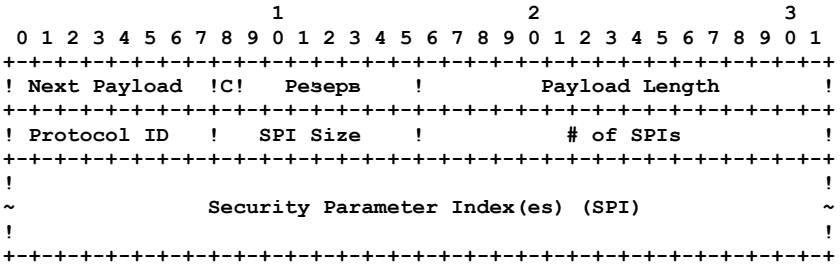

Рисунок 17. Формат удаления.

Удаление IKE_SA указывается значением идентификатора протокола 1 (IKE), а не значениями SPI. Удаление CHILD_SA (таких, как ESP или AH) будет включать идентификатор протокола IPsec (2 для AH, 3 для ESP) и значение SPI, которое передающая точка ожидает во входящих пакетах ESP ил AH.

Элемент Delete включает поля:

- **Protocol ID** (1 октет) - 1 для IKE_SA, 2 для AH, 3 для ESP.
- **SPI Size** (1 октет) - размер (в октетах) SPI, определённого идентификатором протокола. Это поле **должно** иметь нулевое значение для IKE (SPI в заголовке сообщения) или 4 для AH и ESP.
- **# of SPIs** (2 октета) - число SPI в данном элементе Delete. Размер каждого значения SPI определяется полем SPI Size.
- **Security Parameter Index(es)** (переменный размер) - идентифицирует удаляемую защищённую связь (связи). Размер этого поля определяется значением поля SPI Size и числом полей SPI.

Идентификатор типа для элемента Delete имеет значение 42.

3.12. Vendor ID

Элемент данных Vendor ID (далее, V) содержит определённые производителем константы, которые служат для идентификации и распознавания удалённых экземпляров реализаций данного производителя. Этот механизм позволяет производителям экспериментировать с новыми возможностями, обеспечивая совместимость со старыми версиями.

Элемент Vendor ID **может** анонсировать возможность отправителя работать с некими расширениями протокола[1]. **Возможна** передача множества элементов Vendor ID. Реализации **не обязана** передавать элементы Vendor ID и может не использовать их совсем.

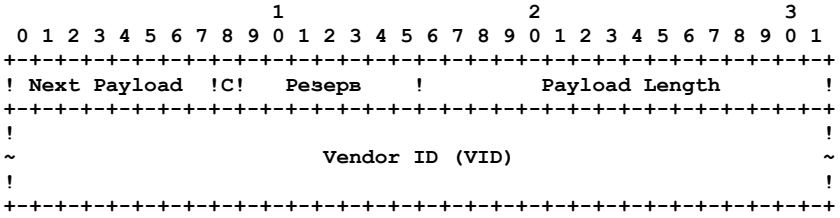

Рисунок 18. Формат элемента Vendor ID.

Элемент Vendor ID может передаваться в любом сообщении. Получение понятного элемента Vendor ID даёт реализации возможность использовать значения, выделенные в этом документе для частного применения - частные элементы данных, типы обмена, уведомления и т. п. Непонятные элементы Vendor ID **должны** игнорироваться. Разработчики проектов протоколов (Internet-Draft), желающие расширить данный протокол, **должны** определить элемент Vendor ID для анонсирования возможности реализовать расширение из Internet-Draft. Предполагается, что получившие признание документы Internet-Draft, будут стандартизоваться с выделением значений из резервных блоков IANA и использование данного Vendor ID будет прекращаться.

Элемент Vendor ID имеет одно поле:

- **Vendor ID** (переменный размер) - несмотря на отсутствие реестра идентификаторов, на выбравшего значение Vendor ID ложится ответственность за уникальность этого идентификатора. Хорошим тоном является включение в идентификатор названия компании, имени разработчика и т. п. Можно также использовать географическую широту и долготу в комбинации со временем выбора значения или придумать иной уникальный идентификатор. Использование цифровой сигнатуры взамен длинных строк является предпочтительной практикой.

Идентификатор типа элемента данных Vendor ID имеет значение 43.

3.13. Элемент данных TS

Элемент данных Traffic Selector (TS) позволяет партнёрам идентифицировать потоки пакетов для обработки службами IPsec.

[1] В оригинале это предложение содержит ошибку, см. https://www.rfc-editor.org/errata_search.php?eid=2194. *Прим. перев.*

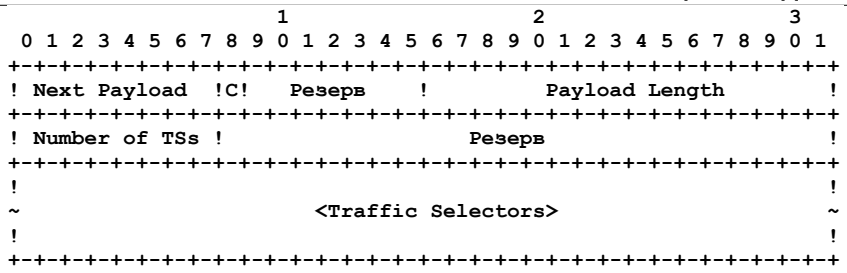

Рисунок 19. Формат элемента данных Traffic Selector.

Элемент TS состоит из базового заголовка IKE, за которым следуют отдельные селекторы трафика.

- **Number of TSs** (1 октет) - число представляемых селекторов трафика.
- **Резерв** - это поле **должно** иметь нулевое значение при передаче и игнорироваться на приёме.
- **Traffic Selectors** (переменный размер) - один или множество индивидуальных селекторов трафика.

Размер элемента TS учитывает заголовок TS и все селекторы трафика.

Идентификатор типа элемента TS имеет значение 44 для адресов на стороне инициатора и 45 для адресов на стороне ответчика.

3.13.1. Субструктура селектора трафика

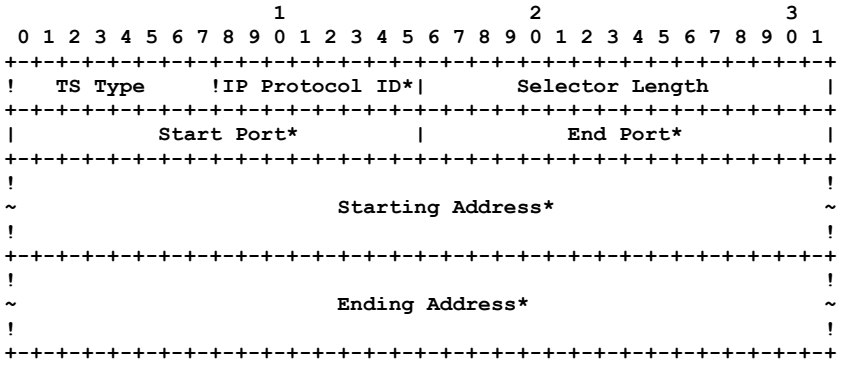

Рисунок 20. Селектор трафика.

** Все поля, за исключением TS Type и Selector Length зависят от значения TS Type. Здесь поля показаны для TS Type 7 и 8 (единственные значения, определённые в настоящий момент).*

- **TS Type** (1 октет) - задаёт тип селектора трафика.
- **IP protocol ID** (1 октет) - значение, показывающее идентификатор связанного протокола IP ID (например, UDP/TCP/ICMP). Нулевое значение говорит о неприменимости идентификатора протокола к данному селектору трафика (SA может передавать все протоколы).
- **Selector Length** - задаёт размер данной субструктуры Traffic Selector с учётом заголовка.
- **Start Port** (2 октета) - задаёт минимальный номер порта, допустимый для данного селектора. Для протоколов, где порт не определён или разрешается использовать любые порты, это поле **должно** иметь значение 0. Для протокола ICMP два однооктетных поля Type и Code трактуются, как 16-битовое целое число (Type занимает старшие 8 битов, а Code - младшие), задающее номер порта для целей фильтрации по данному полю.
- **End Port** (2 октета) - - задаёт максимальный номер порта, допустимый для данного селектора. Для протоколов, где порт не определён или разрешается использовать любые порты, это поле **должно** иметь значение 65535. Для протокола ICMP два однооктетных поля Type и Code трактуются, как 16-битовое целое число (Type занимает старшие 8 битов, а Code - младшие), задающее номер порта для целей фильтрации по данному полю.
- **Starting Address** - Наименьший адрес, включенный в данный селектор (размер определяется полем TS type).
- **Ending Address** - Наибольший адрес, включенный в данный селектор (размер определяется полем TS type).

Системы, соответствующие [RFC4301] и желающие показать все порты (ANY), **должны** устанавливать для начального порта значение 0, а для конечного - 65535 (отметим, что, согласно [RFC4301], ANY включает OPAQUE). Системы, работающие с [RFC4301] и желающие показать порты OPAQUE, но не ANY, **должны** установить для начального порта значение 65535, а для конечного - 0.

В таблице показаны определённые к настоящему моменту значения поля Traffic Selector Type и соответствующие им адресные данные.

TS Type	*Значение*	
Резерв	0 - 6	
TS_IPV4_ADDR_RANGE	7	Диапазон адресов IPv4, представленный двумя четырехоктетными значениями. Первое значение указывает начальный адрес диапазона, второе - конечный. Обе границы и все адреса между ними включаются в диапазон.

TS_IPV6_ADDR_RANGE	8	Диапазон адресов IPv4, представленный двумя шестнадцатиоктетными значениями. Первое значение указывает начальный адрес диапазона, второе - конечный. Обе границы и все адреса между ними включаются в диапазон.
Резерв IANA	9 - 240	
Частное применение	241 - 255	

3.14. Элемент Encrypted[1]

Элемент Encrypted, обозначаемый в этом документе SK{...} или E, содержит другие элементы в зашифрованном виде. При наличии в сообщении элемента Encrypted этот элемент **должен** быть последним в сообщении. Часто этот элемент является единственным элементом данных в сообщении.

Алгоритмы шифрования и защиты целостности согласуются при организации IKE_SA, а ключи рассчитываются в соответствии с параграфами 2.14 и 2.18.

Алгоритмы шифрования и защиты целостности разработаны после алгоритмов ESP, описанных в RFC 2104 [KBC96], 4303 [RFC4303] и 2451 [ESPCBC]. В данном документе полностью описана криптографическая обработка данных IKE, а упомянутые документы следует рассматривать в качестве обоснования устройства протокола. Мы требуем использования блочного шифрования с фиксированным размером блока и алгоритма защиты целостности с фиксированным размером контрольной суммы для сообщений разных размеров.

Идентификатор типа для элемента Encrypted имеет значение 46. Элемент Encrypted включает базовый заголовок IKE и перечисленные ниже поля.

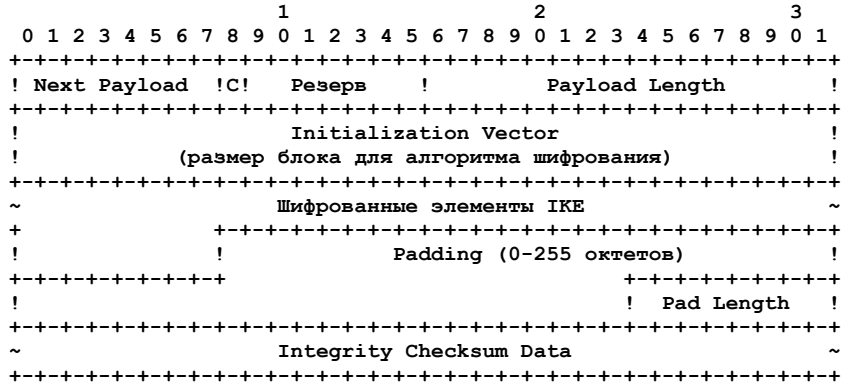

Рисунок 21. Формат зашифрованных данных.

- **Next Payload** - тип первого вложенного элемента данных. Отметим, что это отличается от стандартного формата заголовка IKE - элемент Encrypted является в сообщении последним и, следовательно, значение поля Next Payload в обычных условиях было бы равно 0. Но, поскольку этот элемент содержит в себе другие элементы и нет другого логичного места для указания типа первого элемента, было выбрано это поле.
- **Payload Length** - размер элемента с учётом заголовка IV, Encrypted IKE Payloads, Padding, Pad Length и Integrity Checksum Data.
- **Initialization Vector** - случайное значение, размер которого совпадает с размером блока используемого алгоритма шифрования. Получатели **должны** воспринимать любые значения. Отправителям **следует** следует генерировать псевдослучайное значение независимо для каждого сообщения или использовать для выбора значений шифрованный блок предыдущего сообщения. Для отправителя **недопустимо** использовать одно значение для всех сообщений, последовательности с малым расстоянием Хэмминга[2] (например, порядковые номера) или шифрованные данные из предыдущего принятого сообщения.
- **IKE Payloads** в соответствии с приведённым выше описанием. Это поле шифруется с использованием согласованного алгоритма.
- Поле **Padding может** содержать любое значение, выбранное отправителем и **должно** иметь размер, делающий суммарный размер шифрованных элементов, поля Padding и поля Pad Length кратным размеру блока шифрования. Это поле шифруется с использованием согласованного алгоритма.
- **Pad Length** - размер поля Padding. Отправителю **следует** устанавливать в поле Pad Length минимальное значение, которое делает размер полей шифрованных элементов, Padding и Pad Length кратным размеру блока шифрования, но получатель **должен** принимать любые значения, обеспечивающие требуемое выравнивание. Это поле шифруется с использованием согласованного алгоритма.
- **Integrity Checksum Data** - криптографическая контрольная сумма всего сообщения, начиная с фиксированного заголовка IKE и заканчивая полем Pad Length. Контрольная сумма **должна** рассчитываться для зашифрованного сообщения. Размер поля определяется согласованным алгоритмом защиты целостности.

3.15. Конфигурация

Элемент данных Configuration (CP) используется для обмена конфигурационными параметрами между партнёрами. IKE. Такой обмен включает запрос клиентом IRAC внутреннего адреса IP у сервера IRAS, а также обмен другими данными в процессе получения адреса по протоколу DHCP[3], если клиент IRAC подключён непосредственно к ЛВС.

Элемент Configuration может принимать тип CFG_REQUEST/CFG_REPLY или CFG_SET/CFG_ACK (см. описание поля CFG Type ниже). Элементы CFG_REQUEST и CFG_SET могут добавляться к любому запросу IKE. отклик IKE **должен** включать соответствующий элемент CFG_REPLY, CFG_ACK или уведомление (элемент Notify) с кодом ошибки,

[1]Содержимое этого параграфа частично изменено RFC 5282. *Прим. перев.*
[2]Для строк равной длины дистанцией Хэмминга называют число позиций строк, в которых символы различаются. *Прим. перев.*
[3]Dynamic Host Configuration Protocol - протокол динамической настройки конфигурации хоста.

показывающим причину невозможности выполнения запроса. Исключением являются минимальные реализации, которые **могут** игнорировать все элементы CFG_REQUEST и CFG_SET, поэтому отклик без соответствующего элемента CFG_REPLY или CFG_ACK **должен** приниматься и трактоваться, как индикация того, что запрос не поддерживается.

CFG_REQUEST/CFG_REPLY позволяет конечной точке IKE запросить информацию у партнёра. Если значение атрибута в элементе Configuration типа CFG_REQUEST отлично от 0, такой запрос воспринимается, как предложение для данного атрибута. Элемент Configuration типа CFG_REPLY **может** возвратить это значение или предложить другое. Он **может** также добавить новые атрибуты и не включать некоторые из запрошенных. Запрашивающая сторона **должна** игнорировать атрибуты, которые она не способна распознать.

Некоторые атрибуты **могут** быть многозначными - в этом случае может передаваться и/или возвращаться множество структур Configuration одного типа[1]. В общем случае при запросе атрибута возвращаются все значения. Для некоторых атрибутов (в данной версии спецификации, только для внутренних адресов) множественные запросы показывают, запрос на которому выделение множества значений. Для таких запросов **не следует** возвращать число значений, превышающее запрошенное количество.

Если тип данных, запрошенных в CFG_REQUEST, не распознан или не поддерживается, ответчику **недопустимо** возвращать ошибку - вместо этого он **должен** передать элемент CFG_REPLY, который **может** быть пустым, или вернуть отклик без CFG_REPLY. Возврат ошибки зарезервирован для случаев, когда запрос распознан, но не может быть выполнен должным образом, или запрос имеет некорректный формат.

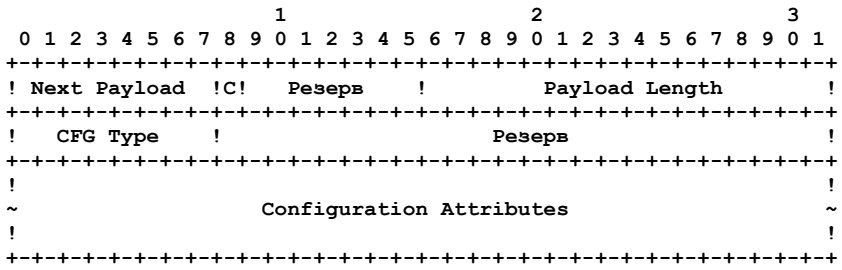

Рисунок 22. Формат конфигурационных данных.

CFG_SET/CFG_ACK позволяет конечной точке IKE «вытолкнуть» конфигурационные данные своему партнёру. В этом случае элемент Configuration типа CFG_SET содержит атрибуты, которые инициатор хочет изменить у своего партнёра. Ответчик **должен** вернуть элемент Configuration, если он принимает какие-нибудь конфигурационные данные. Этот отклик **должен** содержать воспринятые ответчиком атрибуты без данных (данные нулевой длины). Непринятые атрибуты **недопустимо** включать в CFG_ACK Configuration. Если не был принят ни один из атрибутов, ответчик **должен** вернуть пустой элемент типа CFG_ACK или отклик без элемента CFG_ACK. В настоящее время использование обмена CFG_SET/CFG_ACK не определено, хотя такой обмен может применяться в соединениях с расширениями на базе Vendor ID. Минимальные реализации данной спецификации **могут** игнорировать элементы CFG_SET.

Расширения через элемент CP **не следует** использовать для управления общего типа. Основным назначением такого расширения является обеспечение механизма загрузки с обменом информацией IPsec между IRAS и IRAC. Хотя такой подход **может** применяться в качестве метода обмена информацией между защитными шлюзами (SGW[2]) или небольшими сетями, для управления крупными сетями и повторяющихся информационных обменов предпочтительно использовать существующие протоколы управления типа DHCP [DHCP], RADIUS [RADIUS], SNMP или LDAP [LDAP].

Рисунок 22 иллюстрирует формат элемента данных Configuration.

Идентификатор типа элемента Configuration имеет значение 47.

CFG Type	*Значение*
Резерв	0
CFG_REQUEST	1
CFG_REPLY	2
CFG_SET	3
CFG_ACK	4

- **CFG Type** (1 октет) - тип обмена, представленного атрибутами элемента Configuration.

  Определённые значения типов показаны в таблице справа. Значения 5-127 зарезервированы IANA, значения 128-255 выделены для частного применения по согласованию сторон.

- **Резерв** (3 октета) - **должно** устанавливаться в 0 при передаче и игнорироваться на приёме.

- **Configuration Attributes** (переменные размер) - атрибуты конфигурации в формате «тип-размер-значение», описанные ниже. В элементе Configuration может содержаться множество (в том числе, пустое) атрибутов.

3.15.1. Атрибуты конфигурации

- **Резерв** (1 бит) - **должен** устанавливаться в 0 при передаче и игнорироваться на приёме.

[1] В оригинале это предложение содержит ошибку, см. https://www.rfc-editor.org/errata_search.php?eid=2195. *Прим. перев.*
[2] Security Gateway.

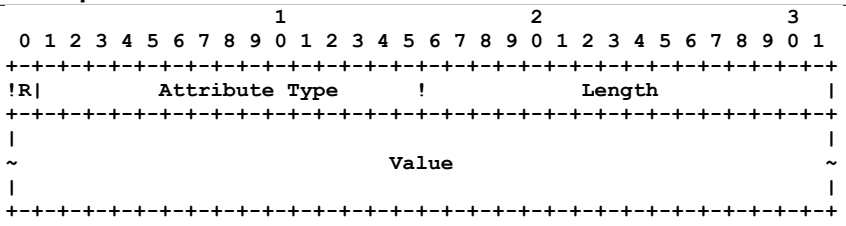

Рисунок 23. Формат атрибута конфигурации.

- **Attribute Type** (15 битов) - уникальный идентификатор для каждого типа атрибутов конфигурации.
- **Length** (2 октета) - размер поля Value в октетах.
- **Value** (0 или более октетов) - поле переменного размера, содержащее значение атрибута конфигурации.

  Определённые в данное время типы атрибутов показаны в таблице справа. Значения типов 16 - 16383 зарезервированы IANA, диапазон значений 16384-32767 выделен для частного применения по согласованию сторон.
- **INTERNAL_IP4_ADDRESS**, **INTERNAL_IP6_ADDRESS** - адрес внутренней сети, иногда называемый адресом красного узла (red node address) или приватным адресом, этот адрес **может** быть приватным с точки зрения сети Internet. Указанный в запросе адрес является запрашиваемым; если в запросе указан 0, это говорит о том, что адрес не запрашивается. Если запрошен конкретный адрес, это скорей всего говорит о том, что ранее существовало соединение с таким адресом и запрашивающий узел хочет снова использовать данный адрес. В IPv6 запрашивающий узел **может** указать младшие байты желаемого адреса. **Можно** запросить множество внутренних адресов, запрашивая множество атрибутов для внутреннего адреса. Ответчик **может** вернуть число адресов, не превышающее запрошенное количество. INTERNAL_IP6_ADDRESS может включать до двух полей, первое из которых содержит шестнадцатиоктетный адрес IPv6, а второе - однооктетный размер префикса, как определено в [ADDRIPV6].

  Запрашиваемый адрес остаётся корректным до окончания срока его действия, заданного атрибутом INTERNAL_ADDRESS EXPIRY, или до завершения всех IKE_SA между партнёрами.
- **INTERNAL_IP4_NETMASK** - маска внутренней сети. В запросах и откликах может содержаться только одна маска (например, 255.255.255.0) и она **должна** использоваться только с атрибутом INTERNAL_IP4_ADDRESS.

Тип атрибута	*Значение*	*Многозначный*	*Размер*
Резерв	0		
INTERNAL_IP4_ADDRESS	1	Да[1]	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[1]	0 или 17 октетов
Резерв	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	кратно 2
INTERNAL_IP6_SUBNET	15	Да	17 октетов

- **INTERNAL_IP4_DNS**, **INTERNAL_IP6_DNS** - задаёт адрес сервера DNS внутри сети. **Можно** запрашивать адреса множества серверов DNS. Ответчик **может** возвращать 0 или более атрибутов серверов DNS.
- **INTERNAL_IP4_NBNS**, **INTERNAL_IP6_NBNS** - задаёт адрес сервера имён NetBios (WINS) внутри сети. **Можно** запрашивать адреса множества серверов NBNS. Ответчик **может** возвращать 0 или более атрибутов серверов NBNS.
- **INTERNAL_ADDRESS_EXPIRY** - задаёт время (в секундах), в течение которого хост может использовать внутренний адрес IP. Хост **должен** обновить IP-адрес до завершения срока его аренды. В отклике **может** присутствовать только один такой атрибут.
- **INTERNAL_IP4_DHCP**, **INTERNAL_IP6_DHCP** - говорит хосту о необходимости отправки всех внутренних запросов DHCP по адресу, содержащемуся в этом атрибуте. **Можно** запрашивать адреса множества серверов DHCP. Ответчик **может** возвращать 0 или более атрибутов серверов DHCP.
- **APPLICATION_VERSION** - версия или информация приложения хоста IPsec, заданная в форме строки печатаемых символов ASCII **без** null-символа завершения.
- **INTERNAL_IP4_SUBNET** - защищаемые данным краевым устройством подсети. Атрибут может содержать до двух полей, первое из которых задаёт адрес IP, а второе - маску. **Можно** запрашивать множество подсетей. Ответчик **может** возвращать 0 или более атрибутов подсетей.
- **SUPPORTED_ATTRIBUTES** - при использовании в запросе этот атрибут **должен** иметь нулевой размер и задаёт запрос ответчику на получение всех поддерживаемых им атрибутов. Отклик содержит атрибут, который включает набор идентификаторов атрибутов (по 2 октета в каждом). Размер данного атрибута, поделённый на 2 (октета) будет показывать число включённых в отклик поддерживаемых атрибутов.

[1]Эти атрибуты возвращаются многозначными лишь в тех случаях, когда было запрошено множество значений.

- **INTERNAL_IP6_SUBNET** - защищённая данным краевым устройством подсеть. Этот атрибут может содержать до двух полей, первое из которых задаёт шестнадцатиоктетный адрес IPv6, а второй одноктетный размер префикса, как определено в [ADDRIPV6]. **Можно** запрашивать множество подсетей. Ответчик **может** возвращать 0 или более атрибутов подсетей.

Отметим, что в этом документе не даётся рекомендаций по выбору информации, которую реализация будет возвращать в откликах. В частности, мы не даём каких-либо конкретных рекомендаций в части определения сервером IRAS сервера DNS, адрес которого следует возвращать по запросам IRAC.

3.16. Элемент EAP

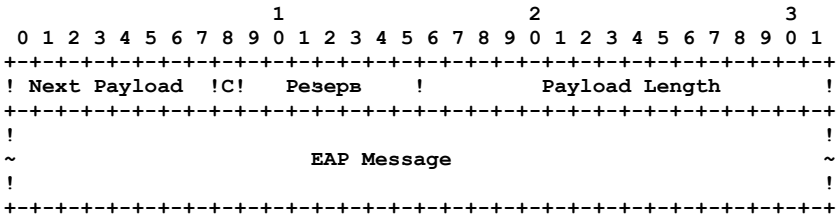

Рисунок 24. Формат EAP.

Элемент Extensible Authentication Protocol[1] (EAP) позволяет выполнять аутентификацию IKE_SA с использованием протокола, определённого в RFC 3748 [EAP] и его последующих расширений. Полный набор приемлемых значений выходит за пределы этой спецификации, однако краткий перечень значений из RFC 3748 включён в документ для использования в наиболее общих случаях.

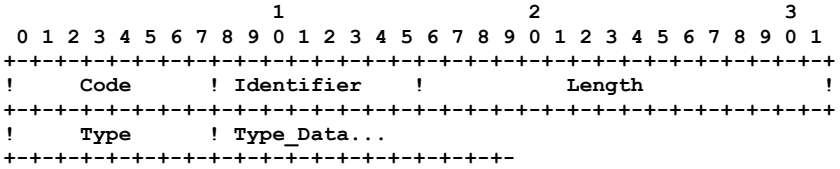

Рисунок 25. Формат сообщения EAP.

Идентификатор типа для элемента данных EAP имеет значение 48.

- **Code** (1 октет) показывает, является это сообщение запросом (Request - 1), откликом (Response - 2), успешным завершением (Success - 3) или отказом (Failure - 4).

- **Identifier** (1 октет) используется в PPP, чтобы отличить повторное использование сообщений от повторной передачи. Поскольку в IKE протокол EAP работает на базе протокола с гарантированной доставкой, использование идентификатора смысла не имеет. В откликах этот октет **должен** устанавливаться равным значению идентификатора в соответствующем запросе. В остальных сообщениях это поле может принимать любое значение.

- **Length** (2 октета) показывает размер сообщения EAP и **должно** быть на 4 меньше значения поля Payload Length инкапсулирующего элемента.

- **Type** (1 октет) присутствует только в сообщениях с кодом Request (1) или Response (2). Для других кодов размер сообщения EAP **должен** составлять четыре октета, а поля Type и Type_Data **должны** отсутствовать. В запросах (1) поле Type показывает запрашиваемые данные, а в откликах (2) поле Type **должно** быть пустым или соответствовать типу запрошенных данных В RFC 3748 определены следующие типы:

  1 Identity - тождественность.

  2 Notification - уведомление.

  3 Nak (только отклики)

  4 MD5-Challenge

  5 One-Time Password (OTP) - однократный пароль.

  6 Generic Token Card - маркерная карта базового типа.

- **Type_Data** (переменный размер) зависит от типа запроса и связанного с ним отклика. Дополнительная информация по методам EAP приведена в работе [EAP].

Поскольку IKE передаёт идентификацию инициатора в протокольном сообщении с номером 3, ответчику **не следует** передавать запросы EAP Identity. Инициатору **следует**, однако, отвечать на такие запросы при их получении.

4. Требования по соответствию

Для обеспечения взаимодействия всех реализаций IKEv2 вводятся специальные требования «**необходимо** поддерживать» (MUST support) в дополнение к другим требованиям. IKEv2 является протоколом защиты и одной из его основных функций является предоставление возможности организации SA только уполномоченным сторонам. Поэтому конкретная реализация может быть настроена с любыми ограничениями по части алгоритмов и удостоверяющих центров, что вступает в противоречие с всеобщей совместимостью.

Протокол IKEv2 разработан так, чтобы минимальные реализации могли взаимодействовать с полнофункциональными реализациями протокола. Существуют группы необязательных функций, которые реализация может игнорировать, если она их не поддерживает. К таким функциям относятся:

[1]Расширяемый протокол аутентификации.

- возможность согласования SA через NAT и туннелирования полученной ESP SA с использованием UDP;
- возможность запрашивать (и отдавать по запросу) временный адрес IP на удалённой стороне туннеля;
- возможность поддерживать различные типы унаследованной аутентификации;
- возможность поддерживать окна размером более 1;
- возможность организации множества SA (ESP и/или AH) в одной IKE_SA;
- возможность замены ключей SA.

Для обеспечения взаимодействия все реализации должны быть способны разбирать все типы элементов данных (или хотя бы пропускать их) и игнорировать непонятные элементы, если в их заголовках не установлен флаг критичности. При наличии флага критичности в заголовке непонятного элемента все реализации **должны** отвергать сообщения с такими элементами.

Каждая реализация **должна** быть способна выполнять обмен 4 сообщениями IKE_SA_INIT и IKE_AUTH, обеспечивающий организацию двух SA (одна для IKE, одна для ESP и/или AH). Реализации могут работать в режиме «только инициатор» или «только ответчик», если это подходит для используемой платформы. Каждая реализация **должна** быть способна отвечать на обмен INFORMATIONAL, но минимальные реализации **могут** отвечать на любое сообщение INFORMATIONAL пустым откликом INFORMATIONAL (отметим, что в контексте IKE_SA «пустое» сообщение состоит из заголовка IKE, за которым следует элемент Encrypted без вложенных в него элементов). Минимальная реализация **может** поддерживать обмен CREATE_CHILD_SA лишь для случаев, когда запрос распознан, и отвергать его с уведомлением типа NO_ADDITIONAL_SAS. От минимальных реализаций не требуется возможность инициирования обменов CREATE_CHILD_SA или INFORMATIONAL. Когда срок жизни SA истекает (по локальному значению времени жизни или числу переданных октетов), реализация **может** попытаться обновить связь с помощью обмена CREATE_CHILD_SA или удалить (закрыть) SA и создать новую. Если ответчик отвергает запрос CREATE_CHILD_SA с передачей уведомления NO_ADDITIONAL_SAS, реализация **должна** быть способна закрыть старую SA и создать новую.

Реализации не обязаны поддерживать возможность запроса временных адресов IP и отклики на такие запросы. Если реализация поддерживает создание таких запросов, она **должна** включать в сообщение 3 элемент данных CP, содержащий по крайней мере поле типа INTERNAL_IP4_ADDRESS или INTERNAL_IP6_ADDRESS. Все остальные поля являются необязательными. Если реализация поддерживает отклики на такие запросы, она **должна** разбирать элемент CP типа CFG_REQUEST в сообщении 3 и распознавать поля типа INTERNAL_IP4_ADDRESS или INTERNAL_IP6_ADDRESS. Если реализация поддерживает выдачу временных адресов запрошенного типа, она **должна** возвратить элемент CP типа CFG_REPLY, содержащий адрес запрошенного типа. Ответчику **следует** включать все остальные связанные с адресом атрибуты, если они имеются.

Минимальная реализация ответчика IPv4 будет игнорировать все содержимое элемента CP, кроме атрибута INTERNAL_IP4_ADDRESS, и будет отвечать на сообщение с включением адреса и других атрибутов, независимо от того, что запрашивал инициатор.

Минимальная реализация инициатора IPv4 будет генерировать элементы CP, содержащие только атрибут INTERNAL_IP4_ADDRESS и разбирать полученный отклик, игнорируя атрибуты, которые она не может использовать. Единственным атрибутом, который **должна** обрабатывать такая реализация, является атрибут INTERNAL_ADDRESS_EXPIRY[1], ограничивающий время жизни SA, если она не будет обновлена до завершения срока жизни. От минимальной реализации инициатора не требуется поддержка запросов на обновление аренды адреса, а минимальные реализации ответчиков не поддерживают откликов на такие запросы.

Для того, чтобы реализация считалась соответствующей данной спецификации, она **должна** обеспечивать возможность настройки на восприятие:

- сертификатов PKIX, содержащих ключи RSA размером 1024 или 2048 и подписанных с использованием таких ключей, когда передаваемый идентификатор является ID_KEY_ID, ID_FQDN, ID_RFC822_ADDR или ID_DER_ASN1_DN;
- аутентификации с общим ключом, когда передаваемый идентификатор является ID_KEY_ID, ID_FQDN или ID_RFC822_ADDR;
- аутентификации ответчика с использованием сертификатов PKIX, а инициатора - с использованием общего ключа.

5. Вопросы безопасности

Хотя протокол разработан так, чтобы минимизировать раскрытие конфигурационной информации неуполномоченным партнёрам, некоторое раскрытие всё-таки неизбежно. Один из партнёров. в любом случае должен сначала идентифицировать себя и подтвердить эту идентификацию. Для предотвращения зондирования к инициаторам обмена предъявляется требование идентифицировать себя первым и обычно требуется первым подтвердить свою подлинность. Однако инициатор может узнать, что ответчик поддерживает IKE и определить поддерживаемые им криптографические протоколы. Ответчик (или кто-нибудь, прикидывающийся таковым) может не только проверить идентификацию инициатора, но и с помощью элементов CERTREQ определить, какие сертификаты желает использовать инициатор.

Использование аутентификации EAP несколько меняет возможности зондирования. При использовании EAP ответчик подтверждает свою идентификацию раньше инициатора, поэтому любой инициатор, которому известно имя корректного инициатора, может зондировать ответчика для выяснения его имени и сертификатов.

Повторяющаяся замена ключей с использованием CREATE_CHILD_SA без дополнительных обменов Diffie-Hellman делает все SA уязвимыми для криптоанализа одного ключа или перегрузки (overrun) любой из конечных точек.

[1] В дополнение к поддержке атрибута INTERNAL_IP4_ADDRESS. *Прим. перев.*

Разработчикам следует отметить этот факт и установить предел для числа обменов CREATE_CHILD_SA между сторонами. В этом документе данный предел не задаётся.

Стойкость ключей, созданных в результате обмена Diffie-Hellman с использованием любой из определённых здесь групп, зависит от стойкости самой группы, размера используемого показателя и энтропии при генерации случайных значений. По причине большого числа влияющих на стойкость ключей факторов сложно определить стойкость ключа для любой из определённых групп. Группа Diffie-Hellman номер 2 при использовании с качественным генератором случайных чисел и показателем не менее 200 битов является общепринятой для 3DES. Группа 5 обеспечивает лучшую защиту по сравнению с группой 2. Группа 1 сохранена в качестве достояния истории и не обеспечивает достаточной защиты за исключением случаев применения с алгоритмом DES, который тоже стал уже достоянием истории. Разработчикам следует принимать во внимание эти оценки при выборе политики и согласовании параметров защиты.

Отметим, что отмеченные выше ограничения относятся к самим группам Diffie-Hellman. В IKE нет запретов на использование более сильных групп и ничто не снижает уровень стойкости, обеспечиваемый наиболее сильными группами (уровень стойкости ограничивается только выбранными при согласовании алгоритмами, включая функцию prf). Фактически, расширяемая схема IKE поощряет определение новых групп - применение групп эллиптических кривых может существенно повысить уровень стойкости при использовании значительно меньших чисел.

Предполагается, что все показатели Diffie-Hellman удаляются из памяти после использования. В частности, такие показатели **недопустимо** создавать на базе долгоживущих секретов типа «затравок» (seed) генератора случайных чисел, которые не уничтожаются после использования.

Стойкость всех ключей ограничивается размером результата согласованной функции prf. По этой причине функции prf, выходное значение которых имеет размер менее 128 битов (например, 3DES-CBC) **недопустимо** использовать с протоколом IKE.

Безопасность данного протокола критически зависит от уровня хаотичности случайно выбираемых параметров. Случайные значений должны создаваться качественным генератором случайных чисел или источником псевдослучайных чисел с подходящей «затравкой» (см. [RFC4086]). Разработчикам следует обеспечить гарантии того, что использование случайных чисел для создания ключей и значений nonce не может приводить к снижению уровня защиты ключей.

Для информации о причинах выбора многих криптографических параметров протокола рекомендуется обратиться к работам [SIGMA] и [SKEME]. Хотя защита согласованных CHILD_SA не зависит от стойкости алгоритмов шифрования и защиты целостности, выбранных для IKE_SA, реализациям **недопустимо** согласовывать использование NONE в качестве алгоритма защиты целостности IKE или ENCR_NULL в качестве алгоритма шифрования IKE.

При использовании заранее распространённых общих (pre-shared) ключей критически важным становится вопрос уровня хаотичности этих секретов. Жёсткая практика требует обеспечения для этих ключей уровня хаотичности не меньше, чем у самого стойкого из согласуемых ключей. Создание общих ключей из паролей, имён или других источников с малой энтропией не обеспечивает нужной защиты. Такие источники не обеспечивают устойчивости к атакам по словарю и использованию методов социальной психологии, а также к другим атакам.

Уведомления NAT_DETECTION_*_IP содержат хэш адресов и портов для сокрытия внутренней структуры сети IP, расположенной за устройством NAT. Поскольку пространство адресов IPv4 включает лишь 32 бита и обычно очень разрежено, атакующий может найти внутренние адреса, скрытые NAT простым перебором всех возможных адресов IP и сравнением хэш-значений. Номера портов обычно содержат фиксированное значение 500, а значения SPI можно выделить из пакетов. Это снижает число попыток расчёта хэш-значений до 2^{32}. Когда можно обоснованно предположить использование адресов из приватных блоков[1], объем расчётов дополнительно сокращается во много раз. Разработчикам, в результате, не следует полагаться на то, что использование IKE гарантирует отсутствие утечки адресной информации.

При использовании методов аутентификации EAP без генерации общих ключей для защиты последующих элементов данных AUTH становятся возможными некоторые MITM-атаки и атаки с подставными серверами [EAPMITM]. Такие уязвимости возникают при использовании EAP с протоколами, которые не защищены безопасным туннелем. Поскольку EAP является протоколом аутентификации общего назначения, который часто используется в системах с единым входом[2], развёрнутое решение IPsec, которое опирается на аутентификацию EAP без генерации общего ключа (этот метод также называют EAP без генерации ключей), может быть скомпрометировано в результате развёртывания совершенно не связанных с ним приложений, использующих тот же метод EAP без генерации ключей, но не обеспечивающих должной защиты. Отметим, что эта уязвимость не связана только с EAP и может возникать также в других сценариях с повторным использованием инфраструктуры аутентификации. Например, если механизм EAP, используемый IKEv2, основан на маркерных идентификаторах, организатор MITM-атаки может использовать подставной web-сервер, перехватить аутентификационный обмен с использованием маркера и применить результат для организации соединения IKEv2. По этой причине **следует** избегать по возможности использования методов EAP без генерации ключей. При использовании таких методов крайне **следует** применять EAP только в защищённых туннелях, когда инициатор проверяет сертификат ответчика до начала обмена EAP. Разработчикам **следует** описывать уязвимость использования методов EAP без генерации ключей в своих реализациях так, чтобы администраторы при развёртывании решений IPsec осознавали возможные риски.

Реализации, применяющие EAP, **должны** также использовать аутентификацию сервера клиенту на основе открытых ключей до начала обмена EAP даже в тех случаях, когда метод EAP предлагает взаимную аутентификацию сторон. Это позволяет избавиться от дополнительных вариаций протокола IKEv2 и защищает данные EAP от активных атак.

Если сообщения IKEv2 слишком велики и требуется фрагментация на уровне IP, атакующие могут заблокировать завершение обмена путём опустошения ресурсов (заполнения буферов) для сборки фрагментов. Вероятность такого исхода можно минимизировать за счёт использования кодирования «Hash and URL» вместо передачи сертификатов (см. параграф 3.6). Дополнительные меры по снижению рисков обсуждаются в работе [KPS03].

[1] См. RFC 1918. *Прим. перев.*
[2] Single-signon facility.

6. Взаимодействие с IANA

В этом документе определено множество новых типов и значений полей, для которых последующее распределение контролируется IANA.

Агентство IANA создало перечисленные ниже реестры.

- IKEv2 Exchange Types (параграф 3.1)
- IKEv2 Payload Types (параграф 3.2)
- IKEv2 Transform Types[1] (параграф 3.3.2)
  - IKEv2 Transform Attribute Types (параграф 3.3.2)
  - IKEv2 Encryption Transform IDs (параграф 3.3.2)
  - IKEv2 Pseudo-random Function Transform IDs (параграф 3.3.2)
  - IKEv2 Integrity Algorithm Transform IDs (параграф 3.3.2)
  - IKEv2 Diffie-Hellman Transform IDs (параграф 3.3.2)
- IKEv2 Identification Payload ID Types (параграф 3.5)
- IKEv2 Certificate Encodings (параграф 3.6)
- IKEv2 Authentication Method (параграф 3.8)
- IKEv2 Notify Message Types (параграф 3.10.1)
  - IKEv2 Notification IPCOMP Transform IDs (параграф 3.10.1)
- IKEv2 Security Protocol Identifiers (параграф 3.3.1)
- IKEv2 Traffic Selector Types (параграф 3.13.1)
- IKEv2 Configuration Payload CFG Types (параграф 3.15)
- IKEv2 Configuration Payload Attribute Types (параграф 3.15.1)

Изменения и дополнения перечисленных реестров осуществляются после экспертизы (процедура expert review).

7. Благодарности

Этот документ является результатом совместных усилий рабочей группы IPsec. Если бы не было ограничений на количество авторов RFC, следовало бы указать всех перечисленных ниже в алфавитном порядке людей - 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. Множество других людей также внесло свой вклад. Это работы по развитию IKEv1, ISAKMP и IPsec DOI, каждая из которых имеет свой авторский коллектив. Hugh Daniel предложил включить нахождение инициатора (в сообщении 3), задал имя для ответчика и дал имя функции «You Tarzan, Me Jane». David Faucher и Valery Smyzlov помогли усовершенствовать процесс согласования селекторов трафика.

8. Литература

8.1. Нормативные документы

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

[ADDRIPV6] Hinden, R. and S. Deering, "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 H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[ESPCBC] Pereira, R. and R. Adams, "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 D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "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. Дополнительная литература

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

[1] При создании нового типа преобразования для него должен создаваться новый реестр.

[DH] Diffie, W., and Hellman M., "New Directions in Cryptography", IEEE Transactions on Information Theory[1], V. IT-22, n. 6, June 1977.

[DHCP] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[DSS] NIST, "Digital Signature Standard", FIPS 186[2], 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. and D. Carrel, "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, B., Pereira, R., and M. Thomas, "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] Wahl, M., Howes, T., and S Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

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

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

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

[PFKEY] McDonald, D., Metz, C., and B. Phan, "PF_KEY 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 R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "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 W. Weiss, "An Architecture for Differentiated Service", RFC 2475, December 1998.

[RFC2522] Karn, P. and W. Simpson, "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 3439, December 2002.

[RFC3715] Aboba, B. and W. Dixon, "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[3], v. 21, n. 2, February 1978.

[SHA] NIST, "Secure Hash Standard", FIPS 180-1[4], 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.

[1]Копия этой статьи доступна на сайте http://www.cs.berkeley.edu/~christos/classics/diffiehellman.pdf. *Прим. перев.*
[2]Копия этого стандарта доступна на сайте http://www.itl.nist.gov/fipspubs/fip186.htm. *Прим. перев.*
[3]Копия этой статьи доступна на сайте http://people.csail.mit.edu/rivest/Rsapaper.pdf. *Прим. перев.*
[4]Копия этого стандарта доступна на сайте http://www.itl.nist.gov/fipspubs/fip180-1.htm. *Прим. перев.*

[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[1].

[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 A: Список отличий от IKEv1

Задачами этого пересмотра IKE явились:

1) определение протокола IKE в едином документе, замена RFC 2407, 2408, 2409 и включение последующих изменений в части добавления работы через NAT, расширяемой аутентификации (EAP) и получения удалённых адресов;
2) упрощение IKE за счёт замены 8 разных начальных обменов одним обменом из 4 сообщений (с изменением механизмов аутентификации, воздействующих только на один элемент AUTH вместо реструктуризации всего обмена), см. [PK01];
3) удаление полей области интерпретации (DOI), ситуации (SIT) и Labeled Domain Identifier, а также битов Commit и Authentication;
4) снижение задержки IKE в общем случае за счёт сведения изначального обмена к 2 периодам кругового обхода (4 сообщения) и разрешения организации CHILD_SA в этом обмене;
5) замена криптографического синтаксиса для защиты самих сообщений IKE синтаксисом, близким к ESP, для упрощения реализации и анализа защиты;
6) снижение числа возможных ошибочных состояний за счёт обеспечения в протоколе гарантий доставки (все сообщения подтверждаются) и упорядочивания, позволивших сократить обмены CREATE_CHILD_SA с 3 сообщений до 2;
7) повышение отказоустойчивости за счёт предоставления ответчику возможности не выполнять существенной обработки до подтверждения инициатором возможности приёма сообщений по заявленному им адресу IP и не менять состояния обмена, пока инициатор не будет криптографически аутентифицирован;
8) устранение криптографических недостатков типа проблем с симметрией в хэш-значениях, используемых для аутентификации (отмечена Tero Kivinen);
9) задание селекторов трафика в специальных элементах данных вместо перегрузки информацией элементов ID и более гибкое указание селекторов трафика;
10) описание поведения при возникновении некоторых ошибок или получении непонятных данных для упрощения совместимости с будущими версиями без ухудшения совместимости с предшествующими;
11) упрощение и прояснение поддержки общих состояний при возникновении ошибок в сети или DoS-атаках;
12) поддержка существующего синтаксиса и «магических» значений для упрощения поддержки в реализациях IKEv1 расширения для работы с IKEv2 при минимальных затратах.

Приложение B: Группы Diffie-Hellman

Имеется две группы Diffie-Hellman, определённых здесь для использования в протоколе IKE. Эти группы создал Richard Schroeppel из Университета штата Аризона. Свойства этих простых чисел описаны в работе [Orm96].

Обеспечиваемой группой 1 стойкости может оказаться недостаточно для обязательных к реализации алгоритмов шифрования и эта группа приведена здесь в силу исторических причин.

Дополнительные группы Diffie-Hellman определены в работе [ADDGROUP].

B.1. Группа 1 - 768-битовая MODP

Этой группе присвоен идентификатор 1.

Простое число имеет значение $2^{768} - 2^{704} - 1 + 2^{64} * \{ [2^{638} \text{ pi}] + 149686 \}$, а его шестнадцатеричная форма имеет вид

```
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 имеет значение 2.

B.2. Группа 2 - 1024-битовая MODP

Этой группе присвоен идентификатор 2.

Простое число имеет значение $2^{1024} - 2^{960} - 1 + 2^{64} * \{ [2^{894} \text{ pi}] + 129093 \}$, а его шестнадцатеричная форма имеет вид

```
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 имеет значение 2.

[1] Копия этой статьи доступна на сайте http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.294&rep=rep1&type=pdf. *Прим. перев.*

Адрес автора

Charlie Kaufman

Microsoft Corporation

1 Microsoft Way

Redmond, WA 98052

Phone: 1-425-707-3335

EMail: charliek@microsoft.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Полное заявление авторских прав

Интеллектуальная собственность

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.