Internet Engineering Task Force (IETF)
Request for Comments: 7296
STD: 79
Obsoletes: 5996
Category: Standards Track
ISSN: 2070-1721

C. Kaufman
Microsoft
P. Hoffman
VPN Consortium
Y. Nir
Check Point
P. Eronen
Independent
T. Kivinen
INSIDE Secure
October 2014

Протокол обмена ключами в Internet версии 2 (IKEv2)

Internet Key Exchange Protocol Version 2 (IKEv2)

Тезисы

В этом документе описана версия 2 протокола IKE[1]. Протокол IKE является компонентой IPsec, используемой для взаимной аутентификации, а также организации и поддержки защищенных связей (SA[2]). Этот документ служит обновлением и заменой RFC 5996 и включает исправления всех обнаруженных там ошибок. Документ заявляет для IKEv2 статус Internet Standard[3].

Статус документа

Этот документ относится к категории Internet Standards Track.

Документ подготовлен IETF[4] и содержит согласованный взгляд сообщества IETF. Документ обсуждался публично и одобрен для публикации IESG[5]. Дополнительная информация о стандартах Internet приведена в разделе 2 RFC 5741.

Информацию о текущем состоянии данного документа, обнаруженных ошибках и способах обратной связи можно найти по ссылке http://www.rfc-editor.org/info/rfc7296.

Оглавление

[1]Internet Key Exchange - обмен ключами в Internet.
[2]Security Association.
[3]STD 79, см. https://www.rfc-editor.org/standards. *Прим. перев.*
[4]Internet Engineering Task Force.
[5]Internet Engineering Steering Group.

1. Введение

IPsec[1] обеспечивает конфиденциальность, целостность данных, контроль доступа и проверку подлинности источника данных для дейтаграмм IP. Эти услуги обеспечиваются за счет поддержки состояния, разделяемого источником и потребителем дейтаграмм IP. Это состояние определяет, наряду с прочим, специфические услуги, обеспечиваемые для дейтаграмм, криптографические алгоритмы, которые будут использоваться для предоставления услуг, а также ключи, применяемые криптоалгоритмами.

Организация такого общего состояния вручную не обеспечивает должного уровня масштабирования. Следовательно, нужен протокол динамической организации таких состояний. В данном документе описывается такой протокол - IKE. Первая версия IKE была определена в RFC 2407 [DOI], 2408 [ISAKMP] и 2409 [IKEV1]. Протокол IKEv2 служит заменой всем этим RFC. Протокол IKEv2 был определен в [IKEV2] (RFC 4306) и дополнительно разъяснен в [Clarif] (RFC 4718). [RFC5996] послужил обновлением и заменой RFC 4306 и RFC 4718. Данный документ заменяет собой RFC 5996. IKEv2, как отмечено в RFC 4306, был заменой протокола IKE, не обеспечивающей совместимости со старой версией. В RFC 5996 документ RFC 4306 был пересмотрен с целью разъяснения IKEv2 с минимальными изменениями, вносимыми в протокол IKEv2. Данный документ заменяет собой RFC 5996, внося некоторые изменения в части продвижения к статусу Internet Standard. Список существенных различий между RFC 4306 и 5996 приведен в параграфе 1.7, а отличий данного документа от RFC 5996 - в параграфе 1.8.

IKE обеспечивает взаимную аутентификацию сторон и организует защищенную связь IKE (SA), включающую общую секретную информацию, которая может использоваться для эффективной организации защищенных связей для ESP[2] [ESP] или AH[3] [AH], а также задает набор криптоалгоритмов, который будет использоваться защищенными связями для передачи трафика между сторонами. В этом документе термин шифронабор (suite или cryptographic suite) служит для обозначения полного множества криптографических алгоритмов, служащих для защиты SA. Инициатор предлагает один или множество шифронаборов, перечисляя поддерживаемые алгоритмы, которые могут комбинироваться в шифронаборы. IKE может также согласовывать использование сжатия IPComp[4] [IP-COMP] для соединений с ESP или AH. Связи SA для ESP или AH, организуемые с использованием IKE SA, будут называться дочерними SA (Child SA).

Все коммуникации IKE осуществляются в формате пар сообщений «запрос-отклик», которые далее будут называться «обменом» (exchange) или «парой запрос-отклик» (request/response pair). Первый обмен сообщениями, организующий IKE SA, называется обменами IKE_SA_INIT и IKE_AUTH, последующие обмены IKE называются CREATE_CHILD_SA или INFORMATIONAL. В общем случае происходит один обмен IKE_SA_INIT и один обмен IKE_AUTH (в сумме четыре сообщения) для организации IKE SA и первой дочерней SA. В исключительных случаях каждый из этих обменов может осуществляться неоднократно. В любом случае все обмены IKE_SA_INIT **должны** завершаться до выполнения каких-либо других обменов, после этого **должны** завершаться все обмены IKE_AUTH, а далее может выполняться любое число обменов CREATE_CHILD_SA и INFORMATIONAL в произвольном порядке. В некоторых сценариях требуется только одна дочерняя SA между конечными точками IPsec и, следовательно, дополнительных обменов не происходит. Последующие обмены могут использоваться для организации дополнительных Child SA между теми же аутентифицированными парами конечных точек для выполнения функций поддержки (housekeeping).

Поток сообщений IKE всегда состоит из запросов, за которыми следуют отклики. Ответственность за обеспечение надежной доставки лежит на запрашивающей стороне (requester). Если отклик не получен в течение заданного времени ожидания, запрашивающая сторона должна повторить запрос (или разорвать соединение).

Первый обмен в сессии IKE (IKE_SA_INIT) согласует параметры защиты IKE SA, передает nonce и значения Diffie-Hellman.

Второй обмен (IKE_AUTH) передает отождествления, обеспечивает соответствующие им секреты и организует SA для первой (зачастую, единственной) дочерней SA AH или ESP (если не возникает отказа при организации AH или ESP Child SA, когда IKE SA организуется без дочерней SA).

Типами последующих обменов являются CREATE_CHILD_SA (создание дочерней SA) и INFORMATIONAL (удаление SA, информация об ошибках и прочие функции housekeeping). На каждый запрос требуется давать отклик. Запрос INFORMATIONAL без данных (кроме пустого поля Encrypted, требуемого синтаксисом) в общем случае используется для проверки жизнеспособности. Такие обмены не могут происходить до завершения начального обмена.

В приведенных далее описаниях предполагается отсутствие ошибок. Изменение потока сообщений для случаев возникновения ошибок рассмотрено в параграфе 2.21.

1.1. Сценарии использования

IKE используется для согласования защищенных связей ESP или AH в разных сценариях, каждый из которых имеет свои требования.

1.1.1. Взаимодействие между защитными шлюзами в туннельном режиме

В этом сценарии ни одна из конечных точек соединения IP не поддерживает IPsec, но сетевые узлы между точками обеспечивают защиту трафика для части пути между точками. Защита прозрачна для конечных точек и основана на обычной маршрутизации с целью пересылки пакетов через конечные точки туннеля для обработки. Каждая из конечных точек туннеля будет анонсировать набор адресов, расположенных «за ней» и пакеты будут передаваться в туннельном режиме, когда адреса реальных конечных точек указываются во внутреннем заголовке IP.

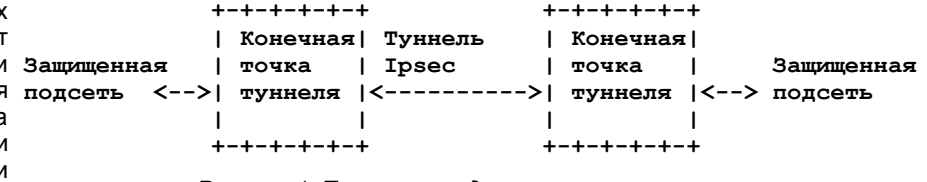


Рисунок 1. Туннель между защитными шлюзами

[1] IP Security - защита IP.
[2] Encapsulating Security Payload.
[3] Authentication Header.
[4] IP Compression.

1.1.2. Взаимодействие между конечными точками в транспортном режиме

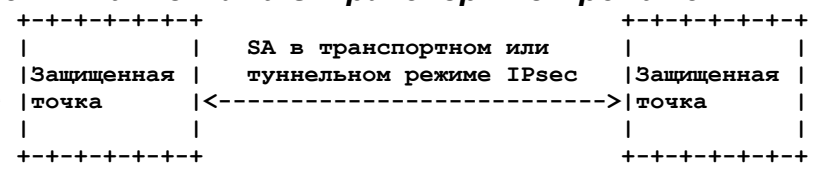


Рисунок 2. Взаимодействие между парой конечных точек

В этом сценарии обе конечных точки соединения IP реализуют IPsec в соответствии с требованиями к хостам [IPSECARCH]. Транспортный режим в общем случае не использует внутреннего заголовка IP. Для пакетов, защищаемых данной SA, согласуется одна пара адресов. Конечные точки **могут** реализовать на прикладном уровне контроль доступа на базе аутентифицированных IPsec отождествлений участников. Этот сценарий обеспечивает сквозную защиту, являющуюся одним из базовых принципов Internet в соответствии с [ARCHPRINC] и [TRANSPARENCY], а также метод снижения унаследованных проблем, связанных со сложностью сетей, как отмечено в [ARCHGUIDEPHIL]. Хотя этот сценарий не полностью применим для IPv4 в Internet, об был успешно развернут в intranet-сетях, использующих IKEv1. Этот метод получит более широкое распространение по мере перехода на IPv6 и адаптации IKEv2.

В этом сценарии возможно размещение одной или обеих защищенных конечных точек за узлом трансляции адресов (NAT[1]). В этом случае туннелируемые пакеты могут инкапсулироваться в UDP так, чтобы номера портов в заголовках UDP могли служить для идентификации конечных точек, расположенных за NAT (см. параграф 2.23).

1.1.3. Туннель между конечной точкой и защитным шлюзом

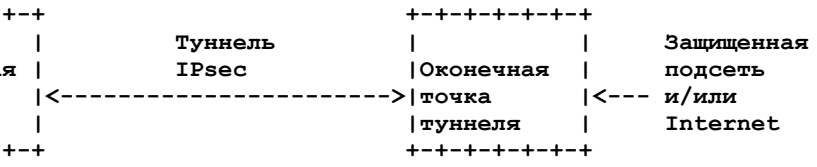


Рисунок 3. Туннель между конечной точкой и защитным шлюзом

В этом сценарии защищенная конечная точка (обычно перемещаемый портативный компьютер) подключается к корпоративной сети через защищенный с помощью IPsec туннель. Этот туннель может использоваться для доступа к ресурсам корпоративной сети или служить для передачи всего трафика через корпоративную сеть с целью использования обеспечиваемых корпоративным межсетевым экраном услуг защиты от атак из сети Internet. В любом случае защищенной конечной точке требуется адрес IP, связанный с защитным шлюзом, чтобы возвращаемые этой точкой пакеты попадали на защитный шлюз и туннелировались назад. Этот адрес IP может задаваться статически или динамически выделяться защитным шлюзом. Для поддержки второго варианта в IKEv2 обеспечивается механизм (а именно, конфигурационные данные - configuration payload), позволяющий инициатору запросить адрес IP, принадлежащий защитному шлюзу для использования в рамках данной SA.

В этом сценарии для пакетов используется туннельный режим. Для каждого пакета от защищенной конечной точки внешний заголовок IP будет содержать IP-адрес отправителя, связанный с текущим местоположением (т. е., адрес, по которому трафик будет маршрутизироваться напрямую к конечной точке), а внутренний заголовок IP будет содержать IP-адрес отправителя, выделенный защитным шлюзом (т. е., адрес, по которому пакеты будут маршрутизироваться защитному шлюзу для туннелирования в конечную точку). В качестве внешнего адреса получателя всегда будет указываться адрес защитного шлюза, а внутренний адрес получателя будет указывать конечного адресата пакета.

В этом сценарии защищенная конечная точка может располагаться за NAT. В таких случаях адрес, видимый защитным шлюзом, не будет совпадать с реальным адресом защищенной конечной точки и пакеты будут инкапсулироваться в UDP для корректной маршрутизации через сеть. Взаимодействие с системами NAT подробно описано в параграфе 2.23.

1.1.4. Другие сценарии

Возможны и другие сценарии, являющиеся вложенными комбинациями описанных выше вариантов. Одним из значимых примеров является комбинация аспектов, описанных в параграфах 1.1.1 и 1.1.3. Подсеть может осуществлять все внешние взаимодействия через удаленный защитный шлюз, используя туннель IPsec, когда адреса подсети маршрутизируются защитному шлюзу через Internet. Примером может служить домашняя сеть, виртуально являющаяся частью Internet со статическими адресами IP, хотя подключение осуществляется через ISP, выделившего один динамический адрес IP для пользовательского защитного шлюза (статические адреса IP и трансляция IPsec обеспечиваются третьей стороной, расположенной в другом месте).

1.2. Начальный обмен

Коммуникации с использованием IKE всегда начинаются с обменов IKE_SA_INIT и IKE_AUTH (в IKEv1 называются фазой 1 - Phase 1). Эти начальные обмены обычно включают 4 сообщения, хотя в некоторых сценариях число этих сообщений может быть больше. Все коммуникации с использованием IKE состоят из пар запрос-отклик. Сначала будет описан базовый обмен, а потом - возможные варианты. Первая пара сообщений (IKE_SA_INIT) согласует криптографические алгоритмы, осуществляет обмен nonce и обмен Diffie-Hellman [DH].

Вторая пара сообщений (IKE_AUTH) аутентифицирует предыдущие сообщения, обменивается отождествлениями и сертификатами, а также организует первую Child SA. Отдельные части этих сообщений шифруются и целостность их защищается с использованием ключей, организованных при обмене IKE_SA_INIT, что позволяет скрыть отождествление от просмотра и аутентифицировать все поля сообщений (в MITM-атаке[2] нарушитель, который не может завершить обмен IKE_AUTH, тем не менее, способен увидеть отождествление инициатора). Информация о генерировании ключей шифрования приведена в параграфе 2.14.

Все сообщения после начального обмена защищаются криптографически с использованием алгоритмов и ключей, согласованных при обмене IKE_SA_INIT. В этих сообщениях используется синтаксис полей данных Encrypted, описанных в параграфе 3.14, которые шифруются с ключами, полученными, как описано в параграфе 2.14. Все последующие сообщения включают поле Encrypted, даже если эти сообщения указаны в документе, как «пустые». Для

[1]Network address translation.
[2]Man-in-the-middle - перехват с участием человека.

обменов CREATE_CHILD_SA, IKE_AUTH и INFORMATIONAL сообщение, следующее после заголовка, шифруется, а целостность сообщения вместе с заголовком защищается с помощью криптоалгоритмов, согласованных для IKE SA.

Каждое сообщение IKE содержит идентификатор Message ID, как часть фиксированного заголовка. Значения Message ID служат для сопоставления запросов и откликов, а также для идентификации повторов передачи сообщений.

В последующих описаниях поля данных (payload), содержащиеся в сообщениях, идентифицируются по именам, приведенным в таблице справа.

Подробные описания всех типов данных (payload) приведены в разделе 3. Данные, являющиеся необязательными, указываются в квадратных скобках - например, [CERTREQ] показывает, что данные запроса сертификата могут включаться опционально.

Обозначение	Тип данных
AUTH	Аутентификация
CERT	Сертификат
CERTREQ	Запрос сертификата
CP	Конфигурация
D	Удаление
EAP	Расширяемая аутентификация
HDR	Заголовок IKE (не данные)
IDi	Идентификация - инициатор
IDr	Идентификация - ответчик
KE	Обмен ключами
Ni, Nr	Nonce
N	Уведомление
SA	Защищенная связь
SK	Зашифровано и аутентифицировано
TSi	Селектор трафика - инициатор
TSr	Селектор трафика - ответчик
V	Идентификатор производителя

Изначальный обмен выполняется следующим образом:

```
   Инициатор                                   Ответчик

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

HDR содержит индексы параметров защиты (SPI[1]), номера версий и флаги разных типов. Поле SAi1 указывает криптографические алгоритмы, которые инициатор поддерживает для IKE SA. Поле KE содержит значение Diffie-Hellman для инициатора, а Ni - nonce инициатора.

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

Отвечающая сторона выбирает шифронабор из предложенных инициатором вариантов и указывает свой выбор в поле SAr1, завершает обмен Diffie-Hellman своим значением в поле KEr и передает свое значение nonce в поле Nr.

На этом этапе согласования каждая из сторон может генерировать значение SKEYSEED (см. параграф 2.14), из которого создаются все ключи для данной IKE SA. Последующие сообщения шифруются и целостность их защищается (за исключением заголовка). Ключи, используемые для шифрования и защиты целостности, создаются из SKEYSEED и называются SK_e (шифрование) и SK_a (аутентификация, иначе защита целостности). Подробное описание создания ключей приведено в параграфах 2.13 и 2.14. Ключи SK_e и SK_a создаются отдельно для каждого направления. В дополнение к ключам SK_e и SK_a, получаемым из значения Diffie-Hellman для защиты IKE SA, создается значение SK_d, используемое при создании ключевого материала для дочерних SA. Нотация SK { ... } показывает, что поля зашифрованы и целостность их защищена с использованием ключей SK_e и SK_a для соответствующего направления.

```
   HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, Sai2, TSi, TSr}  -->
```

Инициатор представляет свое отождествление в поле IDi, подтверждает знание секрета, соответствующего IDi, и защищает целостность содержимого первого сообщения с использованием поля AUTH (см. параграф 2.15). Он может также передать сертификат(ы) в поле(ях) CERT и список доверенных привязок в поле(ях) CERTREQ. При включении какого-либо поля CERT первый представленный сертификат **должен** содержать открытый ключ, используемый для верификации поля AUTH.

Необязательное поле IDr дает инициатору возможность задать отождествления ответчика, которые он желает получить. Это полезно в тех случаях, когда на отвечающей машине имеется множество отождествлений для одного адреса IP. Если предложенное инициатором отождествление IDr неприемлемо для ответчика, тот может использовать иное значение IDr для завершения обмена. Если инициатор не воспримет от отвечающей стороны значение IDr, отличающееся от запрошенного, он может закрыть SA после фиксации этого факта.

Селекторы трафика (TSi и TSr) рассматриваются в параграфе 2.9.

Инициатор начинает согласование Child SA, используя поле SAi2. Заключительные поля (начиная с SAi2) рассмотрены в описании обмена CREATE_CHILD_SA.

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

Ответчик представляет свое отождествление в поле IDr и может также передать один или множество сертификатов (сертификат, содержащий открытый ключ для верификации AUTH, снова должен указываться первым), аутентифицирует свое отождествление и защищает целостность второго сообщения с помощью поля AUTH и завершает согласование Child SA дополнительными полями, рассмотренными ниже при описании обмена CREATE_CHILD_SA. Обе стороны обмена IKE_AUTH **должны** удостовериться, что все сигнатуры и коды MAC[2] рассчитаны корректно. Если любая из сторон использует для аутентификации общий секрет, имена в полях ID **должны** соответствовать ключам, используемым для генерации полей AUTH.

Поскольку инициатор передает значение свое Diffie-Hellman в IKE_SA_INIT, он должен предсказать группу Diffie-Hellman, которую ответчик выберет из предложенного им списка поддерживаемых групп. Если предсказание инициатора окажется ошибочным, ответчик передаст поле Notify типа INVALID_KE_PAYLOAD, показывающее выбранную группу. В этом случае инициатор **должен** повторить IKE_SA_INIT с корректной группой Diffie-Hellman. Инициатор **должен** снова предложить полный список подходящих для него шифронаборов, поскольку сообщение об отказе не было аутентифицировано. Отказ от такой практики позволит организовать активные атаки, в результате которых злоумышленник может вынудить конечные точки к выбору не самого сильного из поддерживаемых обеими сторонами шифронаборов.

[1]Security Parameter Index.
[2]Message Authentication Code - код аутентификации сообщения.

Если при создании Child SA в процессе обмена IKE_AUTH возникает тот или иной сбой, связь IKE SA все равно создается, как обычно. Список типов сообщений Notify в обмене IKE_AUTH, которые не препятствуют организации IKE SA, включает, по крайней мере NO_PROPOSAL_CHOSEN, TS_UNACCEPTABLE, SINGLE_PAIR_REQUIRED, INTERNAL_ADDRESS_FAILURE и FAILED_CP_REQUIRED.

Если отказ связан с созданием IKE SA (например, возвращено сообщение об ошибке Notify с типом AUTHENTICATION_FAILED), связь IKE SA не организуется. Отметим, что, несмотря на шифрование и защиту целостности сообщений IKE_AUTH, если получивший такое сообщение Notify об ошибке партнер еще не аутентифицирован другой стороной (или аутентификация по той или иной причине завершилась отказом), эта информация должна восприниматься с осторожностью. Точнее говоря, в предположении, что значение MAC прошло верификацию, будет известно, что отправителем сообщения Notify является отвечающая сторона обмена IKE_SA_INIT, но отождествление отправителя проверить невозможно.

Отметим, что сообщения IKE_AUTH не содержат полей KEi/KEr и Ni/Nr. Таким образом, поля SA в обмене IKE_AUTH не могут включать Transform Type 4 (группа Diffie-Hellman) со значениями, отличными от NONE. Реализациям **следует** опускать всю субструктуру преобразования вместо передачи значения NONE.

1.3. Обмен CREATE_CHILD_SA

Обмен CREATE_CHILD_SA используется для организации новых Child SA и замены ключей сразу для IKE SA и Child SA. Этот обмен включает одну пару «запрос-отклик»; некоторые из функций этого обмена назывались фазой 2 (Phase 2) в обмене IKEv1. Этот обмен **может** инициироваться любой из сторон IKE SA после завершения начального обмена.

Ключи SA меняются путем создания новой связи SA и последующего удаления прежней связи. В этом параграфе описана первая часть замены ключей (создание новых SA), а в параграфе 2.8 рассмотрены механизмы смены ключей, включая перенос трафика из старых SA в новые и удаление старых связей SA. Эти два параграфа следует прочесть совместно для понимания всего процесса смены ключей.

Любая из конечных точек может инициировать обмен CREATE_CHILD_SA, поэтому в данном параграфе инициатором будет называться именно эта точка. Реализация **может** отвергать все запросы CREATE_CHILD_SA в рамках IKE SA.

Запрос CREATE_CHILD_SA **может** включать поле KE для дополнительного обмена Diffie-Hellman, чтобы обеспечить более высокий уровень защиты для Child SA. Ключевой материал для Child SA обеспечивает функция SK_d, организованная при создании IKE SA, значения nonce, обмен которыми осуществляется в процессе CREATE_CHILD_SA, и значение Diffie-Hellman (если поля KE включены в обмен CREATE_CHILD_SA).

Если обмен CREATE_CHILD_SA включает поле KEi, по крайней мере одна из SA **должна** включать группу Diffie-Hellman этого KEi. Группой Diffie-Hellman для KEi **должен** быть элемент группы, который инициатор предполагает приемлемым для ответчика (могут быть предложены дополнительные группы Diffie-Hellman). Если ответчик выбирает другую группу Diffie-Hellman (отличную от NONE), он **должен** отвергнуть запрос и указать предпочтительную группу Diffie-Hellman с помощью INVALID_KE_PAYLOAD Notify. С этим уведомлением связаны два октета данных, указывающих номер приемлемой группы Diffie-Hellman (порядок битов big endian). В случае такого отказа обмен CREATE_CHILD_SA завершается неудачей и инициатор может повторить обмен с предложением группы Diffie-Hellman и KEi из группы, которую ответчик указал в поле INVALID_KE_PAYLOAD Notify.

Ответчик передает уведомление NO_ADDITIONAL_SAS для индикации неприемлемости запроса CREATE_CHILD_SA по причине нежелания ответчика организовывать дополнительные дочерние SA в данной IKE SA. Такое уведомление может служить также для отказа от смены ключей IKE SA. Некоторые минимальные реализации могут воспринимать организацию только одной связи Child SA в контексте начального обмена IKE и отвергать все последующие попытки добавления.

1.3.1. Создание дочерних SA с помощью обмена CREATE_CHILD_SA

Дочерняя связь Child SA может быть создана путем передачи запроса CREATE_CHILD_SA, как показано ниже.

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

Инициатор отправляет предложение(я) SA в поле SA, значение nonce в поле Ni, опционально указывает значение Diffie-Hellman в поле KEi и предлагаемые селекторы трафика для предлагаемой Child SA в полях TSi и TSr.

Отклик CREATE_CHILD_SA для создания новой Child SA имеет вид

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

Ответчик передает отклик (с тем же значением Message ID) с воспринятым предложением в поле SA, nonce в поле Nr и значением Diffie-Hellman в поле KEr, если в запросе было поле KEi и выбранный шифронабор включает данную группу.

Селекторы для трафика, передаваемого через эту связь SA, задаются в полях TS отклика и могут быть подмножеством предложенных инициатором Child SA.

В запрос **может** включаться уведомление USE_TRANSPORT_MODE, которое также включает поле SA, запрашивающее Child SA. Оно запрашивает для создаваемой Child SA использование транспортного режима вместо туннельного[1]. Если запрос принимается, отклик также **должен** включать уведомление типа USE_TRANSPORT_MODE. Если ответчик отвергает запрос, Child SA будет организована с использованием туннельного режима. Если это неприемлемо для инициатора, он **должен** удалить SA.

Уведомление ESP_TFC_PADDING_NOT_SUPPORTED показывает, что передающая конечная точка не будет воспринимать пакеты с заполнением TFC[2] через согласуемую Child SA. Если ни одна из конечных точек не воспринимает заполнение TFC, это уведомление включается как в запрос, так и в отклик. Если уведомление включено только в одно из сообщений, для противоположного направления заполнение TFC может использоваться.

[1]По умолчанию для всех Child SA используется туннельный режим.
[2]Traffic Flow Confidentiality - конфиденциальность потока трафика.

Для контроля фрагментации используется уведомление NON_FIRST_FRAGMENTS_ALSO. Более подробно описание контроля фрагментации приведено в [IPSECARCH]. Обе стороны должны согласовать между собой передачу отличных от первого фрагментов до того, как любая из них начнет передавать такие фрагменты. Передача возможна лишь в тех случаях, когда уведомление NON_FIRST_FRAGMENTS_ALSO включено как в запрос SA, так и в отклик с согласием на организацию связи. Если ответчик не хочет принимать или передавать отличные от первого фрагменты, он просто не включает уведомление NON_FIRST_FRAGMENTS_ALSO в свой отклик, не отвергая создания Child SA.

В сообщение также можно включать уведомление IPCOMP_SUPPORTED, описанное в параграфе 2.22.

При неудачной попытке создания Child SA **не следует** разрывать организованную связь IKE SA (нет причин для отказа от выполненной работы по созданию IKE SA). Сообщения об ошибках, которые могут возникать при неудачном создании Child SA, описаны в 2.21.

1.3.2. Смена ключей IKE SA с помощью обмена CREATE_CHILD_SA

Запрос CREATE_CHILD_SA для смены ключей IKE SA имеет вид

```
Инициатор                          Ответчик
-----------------------------------------------------------------
HDR, SK {SA, Ni, KEi} -->
```

Инициатор передает предложение(я) SA в поле SA, значение nonce в поле Ni и значение Diffie-Hellman в поле KEi (**должно** включаться). Новое значение SPI инициатора представляется в поле SPI элемента SA. После того, как партнер получит запрос на смену ключей для IKE SA или передаст такой запрос, ему **не следует** начинать обменов CREATE_CHILD_SA для связи IKE SA, в которой будут меняться ключи.

Отклик CREATE_CHILD_SA для замены ключей IKE SA имеет вид

```
                                   <--  HDR, SK {SA, Nr, KEr}
```

Ответчик передает (с тем же значением Message ID) принятое предложение в поле SA, nonce в поле Nr и значение Diffie-Hellman в поле KEr, если выбранный шифронабор включает данную группу. Новое значение SPI ответчика передается в поле SPI элемента SA.

Для новой связи IKE SA значение счетчика сообщений устанавливается в 0, независимо от значения счетчика в предшествующей IKE SA. Первые запросы IKE с обеих сторон новой IKE SA будут иметь Message ID=0. Нумерация старой связи IKE SA сохраняется и любые последующие запросы (например, на удаление IKE SA) будут использовать соответствующие номера. В новой IKE SA также устанавливается размер окна 1, а инициатор смены ключей становится «исходным инициатором» новой связи IKE SA.

В параграфе 2.18 смена ключей IKE SA описана более детально.

1.3.3. Смена ключей Child SA с помощью обмена CREATE_CHILD_SA

Запрос CREATE_CHILD_SA для смены ключей Child SA имеет вид

```
Инициатор                          Ответчик
-----------------------------------------------------------------
HDR, SK {N(REKEY_SA), SA, Ni,
   [Kei], TSi, TSr}             -->
```

Инициатор передает предложение(я) SA в поле SA, значение nonce в поле Ni, опциональное значение Diffie-Hellman в поле KEi и предлагаемые селекторы трафика для предложенной Child SA в полях TSi и TSr.

При обмене для смены ключей могут также передаваться уведомления, описанные в параграфе 1.3.1. Обычно в этом случае применяются те же самые уведомления, которые передавались в исходном обмене (например, для смены ключей SA, работающей в транспортном режиме, передается уведомление USE_TRANSPORT_MODE).

Если целью обмена сообщениями является замена существующей защищенной связи (SA) ESP или AH, в обмен CREATE_CHILD_SA **должно** включаться уведомление REKEY_SA. Связь SA, для которой будут меняться ключи, указывается полем SPI в элементе Notify (это значение SPI, которое инициатор обмена будет ожидать во входящих пакетах ESP или AH). С этим типом сообщений Notify не связано каких-либо данных. Поле Protocol ID в уведомлении REKEY_SA устанавливается в соответствии с протоколом SA, для которой меняются ключи (3 для ESP, 2 для AH).

Отклик CREATE_CHILD_SA для смены ключей Child SA имеет вид

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

Ответчик передает (с тем же Message ID) принятое предложение в поле SA, nonce в Nr и значение Diffie-Hellman в поле KEr, если KEi было указано в запросе и выбранный шифронабор включает эту группу.

Селекторы для трафика, который будет передаваться через данную SA, задаются в полях TS отклика и могут быть подмножеством селекторов, предложенных инициатором Child SA.

1.4. Обмен INFORMATIONAL

В процессе работы IKE SA у партнеров может возникнуть потребность в обмене управляющими сообщениями, связанными с ошибками или некоторыми событиями. Для решения этой задачи в IKE предусмотрен информационный обмен (INFORMATIONAL). Обмены INFORMATIONAL **должны** осуществляться **только** после завершения начальных обменов и защищаться криптографически с использованием согласованных ключей. Отметим, что некоторые информационные сообщения (не обмены), могут передаваться за пределами контекста IKE SA. В параграфе 2.21 более подробно рассматриваются сообщения об ошибках.

Управляющие сообщения, относящиеся к IKE SA, **должны** передаваться через данную IKE SA. Управляющие сообщения, которые относятся к Child SA, **должны** передаваться под защитой IKE SA, в которой создана дочерняя связь (или наследующей IKE SA, если были сменены ключи).

Сообщения в обмене INFORMATIONAL могут содержать поля Notification, Delete и Configuration. Получатель запроса INFORMATIONAL **должен** передать тот или иной отклик. В противном случае отправитель будет предполагать, что

сообщение потерялось в сети и повторно передавать его. Откликом **может** служить пустое сообщение. Запросное сообщение в обмене INFORMATIONAL **может** не содержать никакой информации (payload). Таким способом один из участников может проверить жизнеспособность другой стороны.

Обмен INFORMATIONAL определяется следующим образом

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

Обработка обмена INFORMATIONAL определяется содержимым его компонент.

1.4.1. Удаление SA с помощью INFORMATIONAL

Защищенные связи (SA) ESP и AH всегда существуют парами, по одной SA в каждом направлении. При разрыве SA оба члена пары **должны** быть закрыты (т. е., удалены). Каждая из конечных точек **должна** закрыть свои входящие SA и позволить другой точке закрыть соответствующую SA из каждой пары. Для удаления SA выполняется обмен INFORMATIONAL с одним или множеством полей Delete, указывающих значения SPI (какими они ожидаются во входящих пакетах) удаляемых связей SA. Получатель **должен** закрыть указанные SA. Отметим, что поля Delete никогда не передаются для двух сторон одной SA в одном сообщении. Если одновременно удаляется множество SA, сообщение в обмене INFORMATIONAL включает поля Delete для входящей половины каждой удаляемой пары SA.

Обычно отклики в обмене INFORMATIONAL содержат поля Delete для парных SA противоположного направления, однако имеется одно исключение. Если обе стороны некоторого множества SA независимо решили разорвать эти связи, каждая из сторон может отправить Delete и два запроса могут «столкнуться» в сети. Если узел получает запрос на удаление SA, для которых он сам уже отправил запрос на удаление, он **должен** удалить исходящие SA в процессе обработки запроса, а входящие - в процессе обработки отклика. В таких случаях в отклики **недопустимо** включать поля Delete для удаленных SA, поскольку это приведет к дублированию удаления и может (в теории) привести к удалению не тех SA.

Подобно связям ESP и AH, IKE SA также удаляются с помощью информационного обмена. Удаление IKE SA неявно закрывает все оставшиеся дочерние SA, согласованные для этой связи. Ответ на запрос, который удаляет IKE SA, является пустым откликом INFORMATIONAL.

Полузакрытые соединения ESP и AH являются аномалией и узлам с поддержкой аудита, вероятно следует проверять наличие таких соединений. Отметим, что данная спецификация не задает период ожидания, так что каждая конечная точка вольна самостоятельно определять его. Узел **может** отказаться принимать данные через полузакрытые соединения, но **недопустимо** в одностороннем порядке закрывать их с повторным использованием SPI. Если в состояниях накопилось достаточно путаницы, узел **может** закрыть IKE SA, как описано выше. Нужные SA можно перестроить, организовав новую связь IKE SA.

1.5. Информационные сообщения за пределами IKE SA

Иногда возникают ситуации, когда узел получает пакет, который он не может обработать, и хочет уведомить отправителя об этом.

- Пакет ESP или AH с нераспознанным значением SPI. Это может возникать в результате недавней аварии на принимающем узле с потерей информации о состоянии, а также по причине ошибок в работе других систем или в результате атаки.
- Пакет с зашифрованным запросом IKE приходит в порт 500 или 4500 с нераспознанным значением IKE SPI. Это может возникать в результате недавней аварии на принимающем узле с потерей информации о состоянии, а также по причине ошибок в работе других систем или в результате атаки.
- Пакет с запросом IKE приходит со старшей частью номера версии, значение которой превышает поддерживаемое реализацией.

В первом случае, если принимающий узел имеет активную IKE SA для IP-адреса, с которого был получен пакет, он **может** передать уведомление INVALID_SPI отправителю пакета через IKE SA с использованием обмена INFORMATIONAL. В поле Notification Data указывается значение SPI неприемлемого пакета. Получатель этого уведомления не сможет определить, относится SPI к AH или ESP, но это не имеет существенного значения, поскольку в большинстве случаев значения SPI для этих протоколов будут разными. При отсутствии подходящей IKE SA узел **может** передать информационное сообщение без криптографической защиты по IP-адресу отправителя пакета, используя UDP-порт отправителя в качестве порта назначения, если принятый пакет относился к протоколу UDP (ESP или AH с инкапсуляцией в UDP). В этом случае получателю следует использовать такое сообщение, только как намек на возможное возникновение каких-то неполадок, поскольку такое сообщение очень просто подделать. Это сообщение не является частью обмена INFORMATIONAL и принимающему узлу **недопустимо** отвечать на него, поскольку такой отклик может приводить к зацикливанию обмена сообщениями. В сообщениях этого типа нет значений IKE SPI, важных для получателя, поэтому возможна установка нулевых или случайных значений (исключением является правило параграфа 3.1, которое запрещает использование нулевых значений IKE Initiator SPI). Флаг Initiator имеет значение 1, флаг Response - 0, а флаг версии устанавливается, как обычно (описания флагов приведены в параграфе 3.1).

Для второго и третьего варианта сообщения всегда передаются без криптографической защиты (за пределами IKE SA) и включают уведомление INVALID_IKE_SPI или INVALID_MAJOR_VERSION (без данных). Сообщение является откликом и, таким образом, передается по адресу IP и в порт, откуда поступил запрос, с теми же значениями IKE SPI, Message ID и Exchange Type, которые были в запросе. Флаг Response имеет значение 1, флаги версии устанавливаются, как обычно.

1.6. Уровни требований

Определения основных терминов (таких, как защищенная связь - SA), используемых в этом документе, можно найти в [IPSECARCH]. Следует отметить, что в некоторых случаях IKEv2 работает на основе правил [IPSECARCH], как указано в соответствующих разделах этого документа.

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

1.7. Существенные различия между RFC 4306 и RFC 5996

Этот документ содержит разъяснения и уточнения для протокола IKEv2 [IKEV2]. Многие из разъяснений основаны на работе [Clarif]. Изменения, перечисленные в этом документе, обсуждались в рабочей группе IPsec, а после ее расформирования - в списке рассылки IPsec. Документ содержит подробные разъяснения областей, которые оставались неясными в спецификации IKEv2 и, таким образом, будет полезен разработчикам решений IKEv2.

Протокол, описанный в этом документе, сохраняет значения старшего (2) и младшего (0) номера версии, которые использовались в RFC 4306. Таким образом, номер версии **не** отличается от RFC 4306. Предполагается, что небольшое число технических изменений, внесенных здесь, не повлияет на уже развернутые реализации RFC 4306.

Ссылки и рисунки в этом документе более согласованы, нежели в [IKEV2].

Разработчики IKEv2 отметили, что требования уровня **следует** (SHOULD) в RFC 4306 зачастую неясны и не всегда можно определить соответствие реализации этим требованиям. Отмечено также наличие требования уровня **должно** (MUST), не связанных с интероперабельностью. В данном документе приведены дополнительные разъяснения некоторых из этих требований. Во всех случаях, когда слова **следует** или **должно** не выделены шрифтом, они имеют обычную трактовку и не относятся к требованиям интероперабельности [MUSTSHOULD].

Разработчики IKEv2 (и IKEv1) отметили, что приходится много работать с таблицами кодов параграфа 3.10.1 в RFC 4306. Это приводило к тому, что некоторые разработчики не читали остальные части документа. Значительная часть материала из указанных таблиц была перенесена в соответствующие разделы документа.

Из этого документа удалено обсуждение AH и ESP, включенное в RFC 4306 по ошибке, связанной с задержкой между публикацией RFC 4306 и RFC 4301. IKEv2 базируется в основном на спецификации RFC 4301, которая не включает связки SA (SA bundle), присутствовавшие в RFC 2401. Хотя один пакет может проходить обработку IPsec неоднократно, в каждом из таких проходов используется отдельная связь SA и проходы контролируются таблицами пересылки. В IKEv2 каждая из таких SA создается с использованием отдельного обмена CREATE_CHILD_SA.

Из этого документа удалено обсуждение конфигурационного атрибута INTERNAL_ADDRESS_EXPIRY, поскольку его реализация была весьма проблематичной. Реализации, соответствующие этому документу, **должны** игнорировать предложения с конфигурационным атрибутом типа 5 (старое значение для INTERNAL_ADDRESS_EXPIRY). Этот документ также удаляет конфигурационный атрибут INTERNAL_IP6_NBNS.

Этот документ отменяет разрешение отвергать сообщения, в котором элементы данных (payload) располагались в «некорректном» порядке; сейчас реализациям **недопустимо** так поступать. Это обусловлено недостаточной ясностью описания порядка следования элементов.

Списки элементов из RFC 4306 для включения в реестр IANA были усечены с включением лишь тех элементов, которые были действительно определены в RFC 4306. Кроме того, многим таким спискам сейчас предшествуют важные инструкции для разработчиков, в которых указано, что на практике следует обращаться к реестру IANA на момент выполнения разработки, поскольку с момента публикации RFC 4306 были добавлены новые элементы.

В этом документе разъяснено, когда следует передавать зашифрованные и незашифрованные уведомления с учетом текущего состояния согласования.

В этом документе приведено дополнительное рассмотрение процесса согласования комбинированных шифров.

В параграфе 1.3.2 требование «KEi **следует** включать» заменено требованием «**должно** включаться». Это привело также к изменениям в параграфе 2.18.

В параграфе 2.1 добавлен материал об использовании значений SPI и/или IP инициатора для того, чтобы отличать «полуоткрытые» IKE SA от новых запросов.

В этом документе разъяснено использование флага критичности (параграф 2.5).

В параграфе 2.8 предложение «Отметим, что при смене ключей новая Child SA **может** иметь селекторы трафика и алгоритмы, отличные от значений для прежней связи.» заменено на «Отметим, что при смене ключей новой Child SA **не следует** иметь селекторы трафика и алгоритмы, отличные от значений для прежней связи.».

Новый параграф 2.8.2 посвящен одновременной смене ключей IKE SA.

В параграфе 2.13 добавлено ограничение, в соответствии с которым все псевдослучайные функции (PRF[1]), используемые с IKEv2, **должны** иметь ключи переменного размера. Это требование не должно повлиять на существующие реализации, поскольку неизвестно ни одной стандартизованной PRF с фиксированным размером ключа.

Параграф 2.18 требует выполнения обмена Diffie-Hellman при смене ключей IKE_SA. Теоретически, RFC 4306 разрешает выполнять обмен Diffie-Hellman опционально, но это не полезно (или не приемлемо) при смене ключей IKE_SA.

Параграф 2.21 существенно расширен в части рассмотрения различных случаев, когда требуются отклики на ошибки.

В параграфе 2.23 разъяснено, что при работе через NAT пакеты IPsec, с инкапсуляцией в UDP и без нее, должны быть понятными при получении.

[1]Pseudorandom function.

Добавлен параграф 2.23.1, описывающий работу через NAT при запросе транспортного режима.

Добавлен параграф 2.25, объясняющий действия при возникновении конфликтов синхронизации во время удаления и/или смены ключей SA и определяющий два новых уведомления об ошибках (TEMPORARY_FAILURE и CHILD_SA_NOT_FOUND).

В параграф 3.6 добавлен текст: «Реализации **должны** поддерживать метод HTTP для поиска hash-and-URL. Поведение других URL-методов в настоящее время не задается и такие методы **не следует** применять при отсутствии спецификаций для них.».

В параграф 3.15.3 добавлена ссылка на новый документ, относящийся к настройке адресов IPv6.

Приложение C расширено и сделано более понятным.

1.8. Отличия от RFC 5996

В разделах «Тезисы» и «Статус документа» разъяснено, что документ имеет статус Internet Standard.

Добавлен параграф 2.9.2 с описанием селекторов трафика во время смены ключей.

Добавлено упоминание RFC 6989 по вопросу повторного использования показателей Diffie-Hellman (параграф 2.12).

Добавлен элемент Last Substruc в заголовок Proposal Substructure и Transform Substructure (параграфы 3.3.1 и 3.3.2) со значением 0 для последней и 2 или 3 для субструктуры, за которой следуют другие субструктуры.

Добавлена ссылка на RFC 6989 для случая использования групп, отличных от Sophie Germain Modular Exponentiation (MODP) (параграф 3.3.2).

Добавлена ссылка на RFC 4945 в параграфе «Ошибка: источник перёкрестной ссылки не найден».

Признаны устаревшими открытые ключи Raw RSA в параграфе 3.6. В настоящее время продолжается работа по добавлению более общего формата для неразобранных (raw) открытых ключей.

Исправлены параграфы 3.6 и 3.10 в соответствии с сообщениями об ошибках в RFC 5996 (Errata ID 2707 и 3036).

Добавлено замечание в раздел «6. Согласование с IANA» об отказе от использования ключей Raw RSA Key и удалены старые рекомендации (которые уже были выполнены при обработке RFC 5996). Добавлено замечание о том, что агентству IANA следует заменить все упоминания RFC 5996 ссылками на этот документ.

2. Детали и варианты протокола IKE

IKE обычно принимает (слушает) и передает пакеты через порт UDP 500, хотя сообщения IKE могут приниматься и на порту UDP 4500 с использованием слегка отличающегося формата (см. параграф 2.23). Поскольку протокол UDP работает на основе дейтаграмм (без гарантий доставки), IKE включает восстановление при ошибках передачи, включая потерю и повтор пакетов, а также поддельные пакеты. IKE рассчитан на работу при выполнении двух минимальных требований - (1) по крайней мере один из серии повторных пакетов достигнет адресата до тайм-аута и (2) канал не заполнен поддельными и повторно используемыми пакетами до такой степени, что недостаточно пропускной способности сети или ресурсов CPU какой-либо из конечных точек. Даже при невыполнении этих минимальных требований IKE рассчитан на корректное прерывание работы (как при «обрыве» сети).

Хотя сообщения IKEv2 предполагаются короткими, они содержат структуры, для которых не установлен жестко верхний предел размера (в частности, цифровые сертификаты), а IKEv2, сам по себе, не включает механизма фрагментации больших сообщений. Протокол IP определяет механизм для фрагментации сообщений UDP избыточного размера, но реализации существенно различаются по значениям максимального поддерживаемого размера. Кроме того, фрагментация на уровне IP открывает реализации для DoS-атак [DOSUDPPROT]. Более того, некоторые трансляторы NAT и/или межсетевые экраны могут блокировать фрагменты IP.

Все реализации IKEv2 **должны** обеспечивать возможность передачи, приема и обработки сообщений IKE размером до 1280 октетов, а также **следует** обеспечивать возможность передачи, приема и обработки пакетов размером до 3000 октетов. Реализации IKEv2 должны быть осведомлены о максимальном поддерживаемом размере сообщений UDP и **могут** укорачивать сообщения, исключая некоторые сертификаты или шифронаборы, если это позволит сохранить размер сообщения в допустимых пределах. Использование формата «Hash and URL[1]» вместо включения сертификатов позволяет избавиться от большинства проблем. При реализации и настройке следует принимать во внимание, что в условиях, когда поиск URL становится возможным только после организации Child SA, этот метод может не сработать.

Данные UDP во всех пакетах, содержащих сообщения IKE, переданные в порт 4500, **должны** начинаться с префикса из четырех нулей (в противном случае получатель не будет знать, как их обрабатывать).

2.1. Использование таймеров повтора передачи

Все сообщения IKE существуют попарно - запрос и отклик. Организация IKE SA обычно включает два обмена. После организации IKE SA любая из сторон защищенной связи SA может в любой момент инициировать запросы, в каждый момент «на лету» может находиться множество запросов и откликов. Каждое сообщение маркируется, как запрос или отклик, и для каждого обмена одна сторона SA является инициатором (initiator), а вторая - ответчиком (responder).

Для каждой пары сообщений IKE инициатор отвечает за повтор передачи при тайм-ауте. Ответчику **недопустимо** передавать повторный отклик до получения повторного запроса. При получении повторного запроса ответчик **должен** игнорировать (не обрабатывать) его и только повторно передавать отправленный ранее отклик. Инициатор **должен** помнить каждый запрос, пока на него не будет получен соответствующий отклик. Ответчик **должен** помнить каждый отклик, пока не будет получен запрос, порядковый номер которого не меньше суммы порядкового номера отклика и размера окна (см. параграф 2.3). В целях экономии памяти ответчикам разрешается забывать переданный отклик по истечении тайм-аута в несколько минут. Если ответчик получает повторный запрос для забытого отклика, он **должен** игнорировать этот запрос (не пытаясь создать новый отклик на него).

[1] Хэш и ссылка на оригинал.

IKE является протоколом с гарантией доставки - инициатор **должен** повторять запрос, пока на него не будет получен соответствующий отклик или не будет принято решение о возникновении сбоя в IKE SA. В последнем случае инициатор отбрасывает все состояния, связанные с IKE SA и всеми Child SA, согласованными в данной IKE SA. Повторные запросы от инициатора **должны** быть идентичны исходному запросу (с точностью до бита). Т. е., все биты, начиная с заголовка IKE (перед SPI инициатора IKE SA) должны совпадать, элементы до заголовка IKE (например, заголовки IP и UDP) могут отличаться в повторах.

Повторная передача запроса IKE_SA_INIT требует специальной обработки. Когда ответчик получает запрос IKE_SA_INIT, он проверяет, является пакет повторным, относящимся к имеющимся «полуоткрытым IKE SA (в этом случае повторяет передачу созданного ранее отклика), или новым (в этом случае ответчик создает новую IKE SA и передает соответствующий отклик) запросом или относится к существующей IKE SA, в которой запрос IKE_AUTH уже был получен (в этом случае ответчик игнорирует запрос).

Для дифференциации трех описанных выше случаев недостаточно значений SPI и/или IP-адреса инициатора, поскольку два разных партнера за одним устройством NAT могут выбрать одинаковые значения SPI для инициатора. Отказоустойчивый ответчик будет выполнять поиск IKE SA, используя весь пакет, его хэш или данные Ni.

Политика повтора односторонних сообщений несколько отличается от политики повтора обычных сообщений. Поскольку подтверждения никогда не передаются, не возникает оснований для повторной передачи односторонних сообщений. С учетом того, что все такие сообщения связаны с ошибками, имеет смысл передавать их однократно для каждого пакета с ошибкой и повторять сообщение только при получении других пакетов с ошибками. Имеет смысл также ограничивать частоту передачи таких сообщений об ошибках.

2.2. Использование порядковых номеров для Message ID

Каждое сообщение IKE содержит идентификатор Message ID в своем фиксированном заголовке. Значение Message ID служит для сопоставления откликов с запросами и обнаружения повторной передачи сообщений. При повторе сообщения **должно** использоваться такое же значение Message ID, как в исходном сообщении.

Message ID представляет собой 32-битовое число, которое имеет нулевое значение для сообщений IKE_SA_INIT (включая повторы по причине откликов типа COOKIE и INVALID_KE_PAYLOAD) и инкрементируется для каждого последующего обмена. Таким образом, первая пара сообщений IKE_AUTH имеет Message ID = 1, вторая пара (при использовании EAP) будет иметь идентификатор 2 и т. д. Значение Message ID сбрасывается в 0 для новой IKE SA после смены ключей IKE SA.

Каждая конечная точка IKE SA поддерживает два «текущих» значения Message ID - следующее значение, которое будет использоваться для инициируемого этой точкой запроса и следующее значение, которое она ожидает увидеть в запросе с другой стороны. Эти счетчики инкрементируются по факту генерации или получения запроса. Отклики всегда содержат значения Message ID из соответствующего запроса. Это значит, что после начального обмена каждое целочисленное значение n может появляться в качестве Message ID четырех разных сообщений - n-го запроса от исходного инициатора IKE, соответствующего отклика, n-го запроса от исходного ответчика IKE и соответствующего отклика. Если две стороны передают существенно различающиеся количества запросов, значения Message ID для каждого из направления будут сильно отличаться. Здесь не возникает какой-либо неоднозначности, поскольку флаги Initiator и Response в заголовках сообщений позволяют идентифицировать каждое из четырех сообщений с совпадающими идентификаторами.

В этом документе термин «инициатор» указывает сторону, которая начинает описываемый обмен сообщениями. Исходным инициатором называется сторона, начавшая обмен, который привел к организации текущей IKE SA. Иными словами, если исходный инициатор начинает смену ключей IKE SA, он остается исходным инициатором и для новой IKE SA.

Отметим, что значения ID защищаются криптографически, что позволяет предотвратить повторное использование сообщений. В маловероятной ситуации, когда значения Message ID перестают помещаться в 32 бита, IKE SA **должна** быть закрыта или ключи сменены.

2.3. Размер окна для перекрывающихся запросов

Уведомление SET_WINDOW_SIZE говорит о том, что передавшая его точка способна сохранять состояния для множества продолжающихся обменов, что позволяет получателю уведомления передавать множество запросов до получения отклика на первый запрос. Данные, связанные с уведомлением SET_WINDOW_SIZE, **должны** иметь размер 4 октета и указывать число (порядок битов big endian) сообщений, которые отправитель уведомления обещает сохранять. Размер окна всегда равен 1 до завершения начального обмена.

Конечная точка IKE **должна** ждать отклика на каждое из своих сообщений прежде, чем передать следующее сообщение, если она не получала от своего партнера уведомления SET_WINDOW_SIZE, показывающего готовность того поддерживать состояния для множества сообщений (это позволяет повысить пропускную способность).

После организации IKE SA для повышения производительности IKE конечная точка IKE **может** вводить множество запросов до получения ответа на какой-либо из них - число незавершенных запросов ограничено установленным партнером значением SET_WINDOW_SIZE. Эти запросы могут передаваться через сеть один за другим. Конечная точка IKE **должна** быть готова к восприятию и обработке запроса при наличии у нее запроса, обработка которого еще не завершена, для того, чтобы в таких ситуациях не возникало блокировки. Конечная точка IKE может также воспринимать и обрабатывать множество запросов при наличии у нее запросов с незавершенной обработкой.

Для конечных точек IKE **недопустимо** превышение заданного партнером размера окна при передаче запросов IKE. Иными словами, если ответчик указал размер окна N, а инициатору нужно отправить запрос X, он **должен** дождаться, пока будут получены отклики на все запросы вплоть до X-N. Конечная точка IKE **должна** сохранять копию (или обеспечивать возможность точного восстановления) каждого переданного запроса, пока не будет получен соответствующий отклик. Конечная точка IKE **должна** сохранять копию (или обеспечивать возможность точного восстановления) для ранее отправленных откликов в количестве не менее заявленного ею размера окна на случай потери отправленных откликов и получения от инициатора повторного запроса.

Конечная точка IKE, поддерживающая размер окна больше 1, должна быть способна обрабатывать входящие запросы без учета порядка их доставки для повышения производительности в условиях отказов в сети или нарушения порядка доставки пакетов.

Размер окна является обычным (возможно, настраиваемым) параметром конкретной реализации и не связан с контролем насыщения (как размер окна TCP). В частности, поведение ответчика при получении уведомления SET_WINDOW_SIZE с размером окна меньше текущего значения не определено. Т. е., в настоящее время не существует способа снижения размера окна для существующих IKE SA, окно можно лишь увеличивать. При смене ключей IKE SA новая IKE SA работает с размером окна, равным 1, пока окно явно не будет увеличено новым уведомлением SET_WINDOW_SIZE.

Уведомление INVALID_MESSAGE_ID передается в случае получения сообщения со значением IKE Message ID, выходящим за пределы поддерживаемого окна. Такие уведомления **недопустимо** передавать в откликах и **недопустимо** подтверждать некорректный запрос. Вместо этого следует информировать другую сторону с помощью обмена INFORMATIONAL, в котором данные Notification содержат 4 октета некорректного значения Message ID. Передача такого уведомления является **необязательной** и скорость отправки таких уведомлений **должна** быть ограничена.

2.4. Синхронизация состояний и тайм-ауты соединений

Конечным точкам IKE можно в любой момент забывать все данные о состоянии IKE SA и соответствующих Child SA. Такое поведение является ожидаемым в случаях аварий или перезагрузки конечной станции. При отказе или повторной инициализации конечной точки важно, чтобы другая сторона могла детектировать это и не расходовать пропускную способность сети на передачу пакетов через разорванные связи SA (отправку в «черную дыру»).

Уведомление INITIAL_CONTACT говорит о том, что данная IKE SA является единственной связью IKE SA между аутентифицированными узлами, активной в данный момент. Оно **может** быть передано, когда IKE SA организуется после аварии и получатель **может** использовать эту информацию для удаления всех остальных IKE SA, имеющихся у него для этого аутентифицированного узла без ожидания тайм-аута. Передача такого уведомления **недопустима** для узла, который может быть реплицированным (например, для корпоративного пользователя в роуминге, которому разрешено одновременное подключение к корпоративному межсетевому экрану из двух разных точек). Если уведомление INITIAL_CONTACT передается, оно **должно** включаться в первый запрос или отклик IKE_AUTH (а не в последующие сообщения), принимающая сторона **может** игнорировать это уведомление в других сообщениях.

Поскольку протокол IKE разработан с учетом возможности DoS-атак из сети, конечным точкам **недопустимо** принимать решение об отказе другой стороны соединения на основании какой-либо маршрутной информации (например, сообщений ICMP) или сообщений IKE, приходящих без криптографической защиты (например, сообщений Notify, говорящих о неизвестных SPI). Конечная точка **должна** принимать решение об отказе другой стороны только после того, как продолжающиеся попытки контакта оставались безответными в течение заданного времени или после получения криптографически защищенного уведомления INITIAL_CONTACT, полученного через другую IKE SA от того же аутентифицированного узла. Конечной точке следует предполагать возможность аварии на другой стороне на основании маршрутной информации и в таких случаях инициировать запрос с целью проверки жизнеспособности другой стороны. Для такой проверки в IKE служат пустые сообщения INFORMATIONAL, на которые (как и на все прочие сообщения IKE) требуется ответ (отметим, что в контексте IKE SA «пустое» сообщение включает заголовок IKE, за которым следует поле Encrypted, не содержащее данных). Если криптографически защищенное (свежее, а не повторное) сообщение было недавно получено от другой стороны, незащищенные сообщения Notify **можно** игнорировать. Реализации **должны** ограничивать скорость, с которой могут предприниматься действия в ответ на получение незащищенных сообщений.

Число попыток и продолжительность ожидания не задаются этой спецификацией, поскольку они не оказывают влияния на интероперабельность. Предлагается повторять попытки не менее нескольких десятков раз в течение периода в несколько минут, но требования разных сред могут отличаться. В соответствии с сетевым этикетом интервалы повтора **должны** увеличиваться экспоненциально, чтобы избежать лавинной загрузки сети и усугубления имеющихся перегрузок. Если во всех SA, связанных с IKE SA, был только исходящий трафик, важно подтвердить жизнеспособность другой стороны во избежание передачи пакетов в «черную дыру». Если через IKE SA или какую-либо из дочерних SA в последнее время не было получено ни одного криптографически защищенного сообщения, следует проверить жизнеспособность другой стороны, чтобы предотвратить отправку сообщений «мертвому» партнеру[1]. Получение свежего криптографически защищенного сообщения в IKE SA или любой из дочерних SA подтверждает жизненность IKE SA и всех дочерних SA. Отметим, что это вносит требования к режимам отказов конечных точек IKE. Реализация должна прекратить передачу через все SA, если тот или иной отказ не позволяет прием через все связанные SA. Если система создает Child SA, которые с точки зрения отказов не зависят от других дочерних связей без возможности связанной IKE SA передавать сообщение Delete, система **должна** согласовывать такие Child SA с использованием разных IKE SA.

Существуют DoS-атаки на инициатора IKE SA, которые можно предотвратить за счет применения на стороне инициатора соответствующих мер. Поскольку два первых сообщения в процессе организации SA не защищены криптографически, атакующий может ответить на сообщение инициатора раньше настоящего ответчика и нарушить процесс организации соединения. Для предотвращения этого инициатор **может** воспринимать множественные ответы на свое первое сообщение, трактуя каждый из них, как потенциально легитимный, отвечать на них и затем отбрасывать некорректные полуоткрытые соединения после получения корректного криптографически защищенного отклика для одного из своих запросов. Когда получен корректный криптографически защищенный отклик, все последующие отклики следует игнорировать, независимо от корректности их криптозащиты.

Отметим, что при выполнении этих правил не возникает необходимости согласования времени жизни SA. Если IKE предполагает, что партнер умер на основе повторяющегося отсутствия откликов на сообщение IKE, удаляется IKE SA и все организованные в ней Child SA.

[1]Такую процедуру иногда называют «детектированием мертвого партнера» (dead peer detection или DPD), хотя в реальности происходит детектирование живого партнера.

Конечная точка IKE может в любой момент удалить неактивные Child SA для освобождения связанных с ними ресурсов. Если конечная точка IKE выбирает удаление Child SA, она **должна** передать данные Delete на другую сторону для уведомления об удалении. Это действует подобно тайм-ауту для IKE SA. Закрытие IKE SA неявно закрывает все связанные Child SA. В этом случае конечной точке IKE **следует** передать данные Delete, указывающие на закрытие IKE SA, если другая сторона больше не отвечает.

2.5. Номера версий и совместимость с новыми версиями

В этом документе описана версия 2.0 протокола IKE - значение старшего номера равно 2, младшего - 0. Данный документ является заменой [IKEV2]. Очевидно, что некоторые реализации захотят поддерживать протоколы версии 1.0 и 2.0, а в будущем и других версий.

Значение старшего номера версии следует увеличивать лишь в тех случаях, когда формат пакетов или требуемые действия меняются столь существенно, что старая версия уже не сможет работать с новой, если будет просто игнорировать непонятные ей поля и выполнять действия заданные старой спецификацией. Младший номер версии показывает новые возможности и **должен** игнорироваться узлами с меньшим значением младшего номера и использоваться с информационными целями узлами с большим значением младшего номера версии. Например, младший номер может показывать возможность обработки недавно определенного типа сообщений Notify. Узел с большим значением младшего номера просто отметит для себя, что его корреспондент с меньшим значением младшего номера не поймет таких уведомлений и, следовательно, их не нужно направлять ему.

Если конечная точка получает сообщение с более высоким, чем у ней, старшим номером версии, она **должна** отбросить сообщение, ей **следует** также передать неаутентифицированное сообщение Notify типа INVALID_MAJOR_VERSION, содержащее максимальный (ближайший) поддерживаемый номер версии. Если конечная точка поддерживает старший номер n, а в уведомлении указана версия m, эта точка **должна** поддерживать все версии от n до m. Если точка получает сообщение с поддерживаемым ею старшим номером версии, она **должна** отвечать с этим же номером. Чтобы предотвратить согласование парой узлов значения старшего номера меньше максимально поддерживаемого обоими узлами, в IKE имеется флаг, показывающий, что узел способен понимать более высокое значение старшего номера версии.

Таким образом, старший номер версии в заголовке IKE показывает версию, использованную для сообщения, а не максимальную поддерживаемую отправителем версию. Если инициатор способен поддерживать версии n, n+1 и n+2, а ответчик - n и n+1, они согласуют между собой версию n+1 и инициатор будет устанавливать флаг поддержки более высокой версии. Если узлы по ошибке (возможно, в результате атаки) согласуют версию n, тогда оба узла будут видеть, что партнер может поддерживать больший номер версии и **должны** разорвать соединение, а потом организовать новое с версией n+1.

Отметим, что это правило не выполняется в IKEv1, поскольку для узлов этого протокола не имеет смысла говорить о возможности поддержки большего номера версии. Поэтому активный атакующий способен вынудить два узла, поддерживающих v2, к согласованию v1. Когда поддерживающий v2 узел переходит в режим v1, ему следует отмечать это в системном журнале.

Для совместимости с будущими версиями значения всех резервных полей (RESERVED) **должны** устанавливаться в 0 реализациями версии 2.0, а на приеме такие реализации **должны** игнорировать содержимое этих полей («будьте консервативны при передаче и либеральны на приеме» [IP]). В этом случае будущие версии протокола смогут использовать резервные поля, заведомо зная, что их содержимое будет игнорироваться не понимающими данное поле версиями. Аналогично, типы данных (payload type), которые не определены, являются резервом на будущее, реализации версии, где эти типы еще не определены, **должны** пропускать такие данные, игнорируя их содержимое.

IKEv2 добавляет флаг критичности (critical) в каждый заголовок данных (payload) для совместимости с будущими версиями. Если для нераспознанных данных установлен флаг критичности, сообщение **должно** быть отвергнуто, а отклик на запрос IKE с такими данными **должен** включать поле Notify типа UNSUPPORTED_CRITICAL_PAYLOAD, указывающее на наличие неподдерживаемого типа данных. В этом элементе данных Notify поле Notification Data содержит 1-октетный тип элемента данных. Если флаг критичности отсутствует для неподдерживаемого типа данных, эти данные **должны** игнорироваться. В элементах данных, передаваемых в откликах IKE, устанавливать флаг критичности **недопустимо**. Отметим, что флаг критичности применим только к типу поля данных, но не к его содержимому. Если тип данных распознан, но поле содержит что-либо непонятное (например, неизвестное преобразование в элементах данных SA или неизвестный тип уведомления в поле Notify), флаг критичности игнорируется.

Хотя новые типы полей, которые могут быть добавлены в новых версиях, могут чередоваться с полями определенных здесь типов, реализациям **следует** передавать определенные в этой спецификации данные в порядке, показанном на рисунках разделов 1 и 2. Реализациям **недопустимо** отвергать, как некорректные, сообщения с иным порядком полей.

2.6. IKE SA SPI и уведомления Cookie

Два первых 8-октетных поля в заголовке пакетов IKE, называемые IKE SPI, служат в качестве идентификаторов соединения в начале пакетов IKE. Каждая конечная точка выбирает одно из двух значений SPI и **должна** выбрать то, которое будет уникальным идентификатором IKE SA. Нулевое значение SPI имеет специальный смысл - оно показывает, что значение SPI удаленной стороны еще не известно отправителю.

Входящие пакеты IKE отображаются на IKE SA только по значениям SPI, а не по значениям (например) IP-адресов отправителей.

В отличие от ESP и AH, где в заголовках сообщений указывается только SPI получателя, в каждом сообщении IKE указывается также SPI отправителя. Поскольку значение SPI, выбранное исходным инициатором IKE SA, всегда указывается первым, конечная точка со множеством открытых IKE SA, желая найти IKE SA по выделенному ей значению SPI, должна использовать флаг Initiator в заголовке для определения в первом или втором поле SPI находится нужное значение.

Для первого сообщения начального обмена IKE инициатор еще не знает SPI отвечающей стороны и будет, следовательно, устанавливать для поля значение 0. Когда обмен IKE_SA_INIT не завершается созданием IKE SA по

причине INVALID_KE_PAYLOAD, NO_PROPOSAL_CHOSEN или COOKIE (см. параграф 2.6), значение SPI ответчика будет нулевым даже в отклике. Однако, если ответчик передает отличное от нуля значение SPI, инициатору не следует отвергать сообщение только по этой причине.

Возможны атаки на IKE путем исчерпания ресурсов состояний и CPU, когда цель атаки забрасывается лавиной запросов инициирования сессий с подставных адресов IP. Эффективность таких атак может быть существенно снижена, если ответчик использует ресурсы CPU по минимуму и не выделяет ресурсов для состояний SA, пока он не знает, что инициатор может получать пакеты, направленные по адресу, с которого был получен запрос.

Когда отвечающая сторона детектирует многочисленные полуоткрытые IKE SA, ей **следует** отвечать на запросы IKE_SA_INIT откликами с уведомлением COOKIE. Размер данных, связанных с таким уведомлением, **должен** составлять от 1 до 64 октетов (включительно), а генерация значения описана ниже. Если отклик IKE_SA_INIT включает уведомление COOKIE, инициатор **должен** повторить запрос IKE_SA_INIT, включив в него уведомление COOKIE, содержащее полученные данные в качестве первого элемента данных и сохранив все прочие элементы неизменными. 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,0), N(INVALID_KE_PAYLOAD)
HDR(A,0), N(COOKIE),
    SAi1, KEi', Ni         -->
                                   <-- HDR(A,B), SAr1, KEr, Nr
```

```
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 никакого влияния на состояние инициатора и ответчика, за исключением того, что осуществляется обмен cookie. В частности, порядковые номера 4 первых сообщений будут иметь нулевые значения, а порядковые номера в двух последних сообщениях будут иметь значение 1. В приведенной схеме обмена сообщениями A - значение SPI, выделенное инициатором, а B - значение SPI, выделенное ответчиком.

Реализация IKE может организовать генерацию значений cookie ответчиком таким образом, что это не будет требовать каких-либо сохраненных состояний для проверки приемлемости cookie при получении второго сообщения IKE_SA_INIT. Алгоритмы и синтаксис генерации cookie не оказывают влияния на интероперабельность реализаций и поэтому не рассматриваются здесь. Ниже приводится пример организации конечной точкой частичной защиты от DoS-атак на основе применения cookie.

Хорошим способом является установка ответчиком значения cookie по следующему алгоритму:

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

где <secret> - случайным образом сгенерированное секретное значение, известное только ответчику и периодически сменяемое, | указывает на конкатенацию. Значение <VersionIDofSecret> следует менять при каждой смене значения <secret>. Значение cookie может быть рассчитано заново при получении повторного запроса IKE_SA_INIT для сравнения полученного значения с расчетным. Если значения соответствуют, ответчик знает, что значение cookie было сгенерировано после смены значения <secret>, а значение IPi должно совпадать с адресом отправителя в первом запросе. Включение SPIi в создание cookie позволяет организовать множество разных значений для множества параллельных IKE SA (предполагается, что инициатор выбирает уникальные SPIi'). Включение Ni в хэш гарантирует, что атакующий, который видел только сообщение 2, не сможет создать корректное сообщение 3. Встраивание в хэш значения SPIi не позволяет атакующему взять значение cookie от другой стороны и инициировать множество обменов IKE_SA_INIT с разными значениями SPI инициатора (и, возможно, разными номерами портов), которые ответчик мог бы принять за множество узлов, расположенных за одним устройством NAT.

Если в процессе организации соединения значение <secret> было изменено, запрос IKE_SA_INIT может быть возвращен с отличным от текущего значением <VersionIDofSecret>. Ответчик в таком случае **может** отвергнуть сообщение, отправив новый отклик с другим значением cookie или **может** сохранять старое значение <secret> в течение некоторого времени после замены и воспринимать значения cookie, созданные с использованием старого секрета. Ответчику не следует воспринимать значения cookie в течение неопределенно долгого периода после замены значения <secret>, поскольку это может ослабить защиту от DoS-атак. Ответчику следует достаточно часто менять значение <secret>, особенно во время атак.

Когда одна сторона получает запрос IKE_SA_INIT со значением cookie, не соответствующим ожидаемому, она **должна** игнорировать это значение и обрабатывать сообщение, как при отсутствии cookie (обычно это заключается в передаче отклика с новым значением cookie). Инициаторам следует ограничивать число обменов cookie, которые они предпринимают (возможно, путем экспоненциального роста интервала). Атакующий может подделать множество откликов с cookie на сообщение инициатора IKE_SA_INIT и каждое такое сообщение с поддельным cookie будет вызывать передачу двух откликов - один от инициатора к ответчику (который будет отвергать cookie), а второй от ответчика к инициатору с корректным значением cookie.

Следует отметить, что термин cookie появился в документе Karn и Simpson [PHOTURIS] при описании управления ключами в IPsec и получил признание. Фиксированные заголовки протокола ISAKMP[1] [ISAKMP] включают два 8-

[1]Internet Security Association and Key Management Protocol - протокол защищенных связей и управления ключами в Internet.

октетных поля cookie и этот синтаксис используется в протоколах IKEv1 и IKEv2, хотя в IKEv2 эти поля названы IKE SPI, а для cookie выделено отдельное поле в элементе данных Notify.

2.6.1. Взаимодействие COOKIE и INVALID_KE_PAYLOAD

Существуют две основных причины повтора инициатором обмена IKE_SA_INIT - запрос ответчиком cookie и желание ответчика использовать не ту группу Diffie-Hellman, которая была включена в данные KEi. Если инициатор получает от ответчика cookie, он должен принять решение о включении значения cookie только в ближайший повтор IKE_SA_INIT или во все последующие повторы.

Если инициатор включает cookie только в один повтор, в некоторых случаях может потребоваться один дополнительный круговой обход. Это требуется также в тех случаях, когда инициатор включает cookie во все повторы, но ответчик этого не поддерживает. Например, если ответчик включает данные KEi в расчет cookie, он будет отвергать запрос, передавая новое значение cookie.

Если оба партнера включают cookie во все повторы, обмен слегка сокращается, как показано на рисунке.

Реализациям **следует** поддерживать такой сокращенный обмен, но **недопустимо** давать отказ реализациям, которые не поддерживают такого сокращенного обмена.

2.7. Согласование криптоалгоритма

Элемент данных типа SA показывают предложение для набора вариантов протоколов IPsec (IKE, ESP, AH) в SA, а также криптоалгоритмов, связанных с каждым протоколом.

Элемент данных SA содержит одно или несколько предложений, каждое из которых включает один протокол. Каждый протокол содержит одно или множество преобразований и каждое из них задает криптографический алгоритм. Преобразование может иметь атрибуты (они требуются только в тех случаях, когда Transform ID не полностью задает криптоалгоритм).

Такая иерархическая структура была разработана для повышения эффективности кодирования предложений для шифронаборов, когда число поддерживаемых наборов велико, поскольку приемлемо множество значений для многочисленных преобразований. Ответчик **должен** выбрать один набор, который может быть подмножеством предложенного в SA с учетом приведенных ниже правил.

Каждое предложение содержит один протокол. Если предложение принимается, поле SA в отклике **должно** содержать этот же протокол. Ответчик **должен** принять одно предложение или отвергнуть все, возвратив сообщение об ошибке. Ошибка указывается типом уведомления NO_PROPOSAL_CHOSEN.

Каждое предложение протокола IPsec содержит не менее одного преобразования, а каждое преобразование содержит тип - Transform Type. Принятый шифронабор **должен** содержать единственное преобразование каждого типа, включенного в предложение. Например, если предложение ESP включает преобразования ENCR_3DES, ENCR_AES с размером ключа 128, ENCR_AES с размером ключа 256, AUTH_HMAC_MD5 и AUTH_HMAC_SHA, принятый набор **должен** содержать одно из преобразований ENCR_ и одно из AUTH_. Таким образом, возможны 6 комбинаций.

Если инициатор предлагает обычные шифры с защитой целостности и комбинированные шифры, требуются два предложения. Одно предложение будет включать все обычные шифры с алгоритмами защиты целостности для них, а второе - комбинированные шифры без контроля целостности (комбинированным шрифтам не разрешается включать какие-либо алгоритмы защиты целостности, кроме NONE).

2.8. Смена ключей

Защищенные связи (SA) IKE, ESP и AH используют секретные ключи, которые следует применять только в течение ограниченного срока для защиты ограниченного объема данных. Это ограничивает время жизни связей SA. Когда срок существования защищенной связи заканчивается, дальнейшее использование SA **недопустимо**. При наличии потребности **может** быть организована новая связь SA. Организация SA заново при завершении времени жизни называется сменой ключей (rekeying).

В интересах минимальных реализаций IPsec возможность смены ключей SA без перезапуска всей IKE SA не является обязательной. Реализация **может** отвергать все запросы CREATE_CHILD_SA в рамках IKE SA. Если срок жизни SA завершился или близок к завершению и попытка смены ключей с помощью описанных здесь механизмов не удается, реализация **должна** закрыть IKE SA и все дочерние связи (Child SA), а после этого **может** организовать новые защищенные связи. Реализации могут пожелать замены ключей SA в процессе работы, поскольку это повышает производительность и вероятно снижает число пакетов, теряемых во время смены ключей.

Для смены ключей Child SA в существующей IKE SA создается новая, эквивалентная связь SA (см. параграф 2.17 ниже), а после ее организации старая связь удаляется. Отметим, что при смене ключей новой Child SA **не следует** иметь селекторов трафика и алгоритмов, отличных от значений для прежней связи.

Для смены ключей IKE SA организуется новая, эквивалентная IKE SA (см. параграф 2.18 ниже) с тем же партнером путем выполнения обмена CREATE_CHILD_SA в рамках существующей IKE SA. Новая связь IKE SA наследует все дочерние связи (Child SA) исходной IKE SA и применяется для всех управляющих сообщений, требуемых для поддержки этих Child SA. После создания новой IKE SA инициатор удаляет старую IKE SA, при этом данные Delete для удаления связи **должны** быть последним запросом, передаваемым через старую IKE SA.

Смену ключей SA следует выполнять заранее, т. е., новую SA следует создавать до того, как у старой завершится время жизни и она станет недоступной. Между созданием новой SA и завершением срока жизни имеющейся связи должен сохраняться интервал, достаточный для переноса трафика в новую SA.

Различие между IKEv1 и IKEv2 заключается в том, что время жизни SA в IKEv1 согласовывалось. В IKEv2 каждая сторона SA отвечает за реализацию своей политики ограничения времени жизни SA и смену ключей SA при необходимости. Если стороны имеют разные правила для времени жизни, смена ключей всегда будет инициироваться стороной с более коротким сроком существования защищенных связей. Если SA была неактивна достаточно долго и если конечная точка не инициировала SA при отсутствии трафика, эта конечная точка может предпочесть закрытие SA

(вместо смены ключей) при завершении срока жизни. Это может происходить также в тех случаях, когда после предшествующей смены ключей трафика через SA не было совсем.

Отметим, что в IKEv2 осознанно разрешена организация параллельных SA с общими селекторами трафика между парой конечных точек. Одна из целей этого заключается в поддержке дифференциации качества обслуживания (QoS[1]) между SA (см. [DIFFSERVFIELD], [DIFFSERVARCH] и параграф 4.1 в [DIFFTUNNEL]). Следовательно, в отличие от IKEv1, комбинация конечных точек и селекторов трафика может не обеспечивать уникальной идентификации SA, организованных между этими точками, поэтому здесь **не следует** применять удаление SA при смене ключей на основании совпадения селекторов трафика, как это принято в IKEv1.

Возможны интервалы времени (в частности, при наличии потерь пакетов), когда стороны не могут договориться о состоянии SA. Ответчик на CREATE_CHILD_SA **должен** быть готов к восприятию сообщений в SA до передачи своего отклика на запрос создания - здесь не возникает двусмысленностей для инициатора. Инициатор **может** начать передачу в SA, как только он обработает отклик. Однако инициатор не может принимать что-либо в новой SA, пока он не получит и не обработает отклик на свой запрос CREATE_CHILD_SA. В таком случая, как ответчик узнает о возможности передачи через созданную SA?

С точки зрения технической корректности и интероперабельности ответчик **может** начать передачу в SA сразу же после отправки отклика на запрос CREATE_CHILD_SA. В некоторых случаях, однако, это может приводить к ненужному отбрасыванию пакетов, поэтому реализация **может** задержать такую передачу.

Ответчик может быть уверен в том, что инициатор готов к приему сообщений через SA, если (1) если он получил криптографически защищенное сообщение на другой половине пары SA или (2) новая SA организована в результате смены ключей существующей SA и был получен запрос IKE на закрытие заменяемой SA. При смене ключей SA ответчик продолжает передавать трафик в старую SA, пока не произойдет одно из указанных выше событий. При создании новой SA ответчик **может** отложить передачу сообщений через нее, пока не получит сообщение через нее или не истечет время ожидания. Если инициатор получает сообщение из SA, для которой он еще не принял отклика на свой запрос CREATE_CHILD_SA, он интерпретирует это, как результат потери пакета и повторяет запрос CREATE_CHILD_SA. Инициатор **может** передать «холостое» (dummy) сообщение ESP в новую ESP SA, если в его очереди нет сообщений для передачи - это позволит ответчику убедиться в том, что инициатор готов к приему сообщений.

2.8.1. Одновременная смена ключей дочерних SA

Если на обеих сторонах используются общие правила для времени жизни, возможно одновременное инициирование смены ключей обеими сторонами (это будет приводить к созданию избыточных SA). Для снижения вероятности таких событий в отправку запросов на смену ключей **следует** вводить варьируемые задержки (задержка на случайное время с момента обнаружения необходимости смены ключей).

Такая форма смены ключей может временно приводить к наличию множества похожих SA между парой узлов. При наличии двух SA, имеющих право получать пакеты, узел **должен** воспринимать пакеты через любую из них. Если в результате конфликта создаются избыточные SA, связь SA, созданную с меньшим из четырех значений nonce, использованных в двух обменах, **следует** закрыть со стороны создавшей эту связь конечной точки. Значения nonce сравниваются пооктетно (вместо сравнения их, как больших целых чисел). Иными словами, сравниваются сначала первые октеты, а при их равенстве вторые и т. д. Если достигнут последний октет одного из значений nonce, это значение будет наименьшим. Узлу, инициировавшему SA, для которой были сменены ключи, следует удалить старую SA после организации новой связи.

Ниже показано влияние описанного на реализации. Предположим, что хосты A и B имеют пару Child SA с SPI (SPIa1,SPIb1) и обе стороны одновременно начинают смену ключей.

```
Хост A                              Хост B
-------------------------------------------------------------------
передача req1: N(REKEY_SA,SPIa1),
    SA(..,SPIa2,..),Ni1,..    -->
                               <--  передача req2: N(REKEY_SA,SPIb1), SA(..,SPIb2,..),Ni2
прием req2                     <--
```

В этот момент A узнает о начавшейся одновременной смене ключей. Однако он еще не знает, который из обменов будет иметь наименьшее значение nonce, поэтому хост просто зафиксирует этот факт и будет отвечать, как обычно.

```
передача resp2: SA(..,SPIa3,..),
                    Nr1,..  -->
                            -->  прием req1
```

В этот момент B также узнает об одновременной смене ключей и отвечает, как обычно.

```
                           <--  передача resp1: SA(..,SPIb3,..), Nr2,..
прием resp1                <--
                           -->  прием resp2
```

В этот момент между хостами будет три Child SA (одна старая и две новых). A и B могут сейчас сравнить значения nonce. Предположим, что меньшим оказалось значение Nr1 из сообщения resp2 - в этом случае B (отправитель req2) удаляет избыточную новую SA, а хост A (инициатор организации SA, для которой менялись ключи) удаляет старую дочернюю связь.

```
передача req3: D(SPIa1)  -->
                         <--  передача req4: D(SPIb2)
                         -->  прием req3
                         <--  передача resp3: D(SPIb1)
прием req4               <--
передача resp4: D(SPIa3) -->
```

Смена ключей на этом завершается.

[1]Quality of service.

Однако возможна иная последовательность событий, если возникнет потеря пакетов в сети, приводящая к повторам передачи. Смена ключей начинается, как обычно, но первый пакет от хоста A (req1) теряется.

```
Хост A                          Хост B
-----------------------------------------------------------------
передача req1: N(REKEY_SA,SPIa1), SA(..,SPIa2,..),
    Ni1,..                  -->  (потеряно)
                            <--  передача req2: N(REKEY_SA,SPIb1), SA(..,SPIb2,..),Ni2
прием req2                  <--
передача resp2: SA(..,SPIa3,..),
    Nr1,..                  -->
                            -->  прием resp2
                            <--  передача req3: D(SPIb1)
прием req3                  <--
передача resp3: D(SPIa1)    -->
                            -->  recv resp3
```

С точки зрения хоста B смена ключей завершена, но, поскольку этот хост еще не получил пакет req1 от хоста A, он еще не знает об одновременной смене ключей. Однако A будет повторять передачу сообщения и оно придет хосту B.

```
повтор req1                 -->
                            -->  прием req1
```

Для B это выглядит, как попытка A сменить ключи для SA, которой уже не существует, поэтому B будет отвечать на запрос сообщением о некритической ошибке (например, CHILD_SA_NOT_FOUND).

```
                            <--  передача resp1: N(CHILD_SA_NOT_FOUND)
прием resp1                 <--
```

Когда A получит это сообщение, он уже будет знать об одновременной смене ключей и может просто игнорировать его.

2.8.2. Одновременная смена ключей IKE SA

Возможно, наиболее сложная ситуация возникает при попытке одновременной смены ключей IKE_SA. Текст параграфа 2.8 применим и к таким ситуациям, однако важно обеспечить дочерним связям SA наследование от корректной IKE_SA.

Ситуация, когда обе стороны видят одновременную замену ключей, обрабатывается так же, как для Child SA. После обмена CREATE_CHILD_SA между хостами A и B будет существовать три IKE SA - старая IKE SA и две новых IKE SA. Новую IKE SA с меньшим значением nonce **следует** удалить создавшему ее узлу, а оставшаяся новая IKE SA **должна** стать родителем всех Child SA.

В дополнение к обычной ситуации одновременной замены ключей существует специальный случай, когда один из партнеров заканчивает процесс смены ключей до того, как узнает о смене ключей другой стороной. Если одновременная смена ключей замечена только одним из партнеров, избыточные SA не создаются. В этом случае, когда не заметивший одновременной смены ключей партнер получает запрос на смену ключей IKE SA, для которой он уже сменил ключи, ему **следует** возвращать сообщение TEMPORARY_FAILURE, поскольку это IKE SA, которую он в настоящее время пытается закрыть (независимо от того, успел ли он передать уведомление о закрытии SA). Если партнер, заметивший одновременную смену ключей, получает от другой стороны запрос на удаление старой IKE SA, он понимает, что партнер не обнаружил одновременной смены и может забыть о своей попытке смены ключей.

В дополнение к обычной ситуации одновременной замены ключей существует специальный случай, когда один из партнеров заканчивает процесс смены ключей до того, как узнает о смене ключей другой стороной. Если одновременная смена ключей замечена только одним из партнеров, избыточные SA не

```
Хост A                              Хост B
-----------------------------------------------------------------
передача req1:
   SA(..,SPIa1,..),Ni1,..    -->
                             <-- передача req2: SA(..,SPIb1,..),Ni2,..
                             --> прием req1
                             <-- передача resp1: SA(..,SPIb2,..),Nr2,..
прием resp1                  <--
передача req3: D()           -->
                             --> прием req3
```

создаются. В этом случае, когда не заметивший одновременной смены ключей партнер получает запрос на смену ключей IKE SA, для которой он уже сменил ключи, ему **следует** возвращать сообщение TEMPORARY_FAILURE, поскольку это IKE SA, которую он в настоящее время пытается закрыть (независимо от того, успел ли он передать уведомление о закрытии SA). Если партнер, заметивший одновременную смену ключей, получает от другой стороны запрос на удаление старой IKE SA, он понимает, что партнер не обнаружил одновременной смены и может забыть о своей попытке смены ключей.

В этот момент хост B видит запрос на закрытие IKE_SA. Тут не требуется ничего, сверх обычного отклика. Однако в этот момент хосту B следует прекратить повтор передачи req2, поскольку с момента получения хостом A отклика resp3 он будет удалять все состояния, связанные со старой IKE_SA и не сможет отвечать на эти запросы.

```
                                    <-- передача resp3: ()
```

Уведомление TEMPORARY_FAILURE не было включено в RFC 4306 и поддержка уведомлений TEMPORARY_FAILURE не согласуется. Таким образом, узлы, реализующие RFC 4306, но не соответствующие этому документу, могут получать эти уведомления. В таких случаях, узлы будут трактовать эти уведомления, как неизвестные сообщения об ошибках и прекращать обмен. Поскольку другой партнер уже завершил смену ключей, это прекращение не будет оказывать какого-либо влияния.

2.8.3. Замена ключей IKE SA по сравнению с повторной аутентификацией

Смена ключей IKE SA и повторная аутентификация концептуально различаются в IKEv2. При смене ключей IKE SA организуются новые ключи для IKE SA и сбрасываются счетчики Message ID, но не выполняется повторной аутентификации сторон (элементы AUTH и EAP не используются).

Хотя смена ключей IKE SA может быть важна для некоторых сред, повторная аутентификация (верификация доступа сторон к долгосрочным свидетельствам) зачастую более важна.

IKEv2 не осуществляет какой-либо специальной поддержки для повторной аутентификации, которая выполняется новой IKE SA с нуля (с использованием обменов IKE_SA_INIT/IKE_AUTH без каких-либо элементов REKEY_SA Notify), создания новых Child SA в созданной IKE SA (без элементов REKEY_SA Notify) и удаления имеющейся IKE SA (при этом удаляются и старые дочерние связи - Child SA).

Это означает, что при повторной аутентификации также создаются новые ключи для IKE SA и Child SA. Следовательно, если менять ключи чаще, чем проводить повторную аутентификацию, ситуации, когда «срок аутентификации» короче времени жизни ключей, не возникают.

Хотя создание новой IKE SA может быть инициировано любой из сторон (инициатором или ответчиком исходной IKE SA), использование элементов EAP и/или Configuration на практике означает, что повторная аутентификация будет инициирована той же стороной, что и для исходной IKE SA. IKEv2 в настоящее время не позволяет ответчику в данном случае запросить повторную аутентификацию, однако существуют расширения, добавляющие такую функциональность [REAUTH].

2.9. Согласование селекторов трафика

Когда соответствующая RFC 4301 подсистема IPsec получает пакет IP, который соответствует селектору protect в базе данных SPD[1], подсистема защищает такой пакет с помощью IPsec. Если SA еще не существует, организация такой связи является задачей IKE. Поддержка SPD системы выходит за пределы IKE, хотя некоторые реализации могут обновлять SPD при работающем IKE (пример сценария приведен в параграфе 1.1.3).

Элементы данных TS[2] позволяют конечным точкам обмениваться с партнерами некоторой информацией из своей SPD. Эти данные должны передаваться в IKE из SPD (например, PF_KEY API [PFKEY] использует сообщения SADB_ACQUIRE). Элементы TS задают критерии отбора для пакетов, которые будут пересылаться через вновь созданную SA. В некоторых сценариях это может служить проверкой согласованности SPD, а в других случаях приводить к динамическому обновлению SPD.

В каждом сообщении обмена, создающего пару Child SA, присутствуют два элемента TS и каждый из них содержит один или множество селекторов трафика. Каждый селектор включает диапазон адресов (IPv4 или IPv6), диапазон портов и идентификатор протокола IP.

Первую пару селекторов трафика обозначают TSi[3], а вторую - TSr[4]. TSi задает адрес отправителя трафика, пересылаемого от инициатора пары Child SA (или получателя трафика, пересылаемого ему). TSr указывает адрес получателя трафика, пересылаемого ответчику пары Child SA (или адрес отправителя трафика, пересылаемого от него). Например, если исходный инициатор запрашивает создание пары Child SA и хочет туннелировать весь трафик из подсети 198.51.100.* на стороне инициатора в подсеть 192.0.2.* на стороне ответчика, инициатор будет включать один селектор трафика в каждое поле TS. TSi будет задавать диапазон адресов (198.51.100.0 - 198.51.100.255), а TSr - диапазон (192.0.2.0 - 192.0.2.255). В предположении, что это предложение приемлемо для ответчика, он будет передавать обратно идентичные элементы TS.

IKEv2 позволяет ответчику выбрать подмножество трафика, предложенного инициатором. Такая потребность может возникать в случае обновления конфигурации конечных точек, когда одна из сторон еще не получила информации об этом обновлении. Поскольку конечные точки могут настраивать разные люди, несогласованность может существовать достаточно долго даже при отсутствии ошибок. Это позволяет также использовать разные конфигурации осознанно, когда одна сторона настраивает туннель для всех адресов и ждет обновления конфигурации с другой стороны.

Когда ответчик выбирает подмножество трафика, предложенного инициатором, он сужает селекторы трафика до некоторого подмножества предложения инициатора (оно не должно стать пустым). Если тип предложенного селектора трафика не известен, ответчик игнорирует селектор, поэтому неизвестный тип не возвращается в суженном наборе.

Чтобы обеспечить ответчику возможность выбора подходящего диапазона в том случае, когда инициатор запрашивает создание SA по причине наличия пакета данных, инициатору **следует** включить в качестве первого селектора трафика в каждое поле TSi и TSr очень специфичный селектор трафика, включающий адреса из вызвавшего запрос пакета. В нашем примере инициатор будет включать в TSi два селектора трафика - первый с диапазоном адресов (198.51.100.43 - 198.51.100.43) , номером порта и протоколом IP из пакета, а второй с диапазоном (198.51.100.0 - 198.51.100.255) для всех портов и протоколов IP. Инициатор будет просто включать в TSr два селектора трафика. Если инициатор создает пару Child SA не в результате получения пакета, а, например, при старте, здесь может не оказаться специфичных адресов, которые инициатор предпочтет для туннеля. В таком случае первые значения в TSi и TSr могут быть диапазонами, а не конкретными значениями.

Ответчик сужает диапазон следующим образом:

- если политика ответчика не позволяет ему принимать никакую часть предложенных селекторов, он отвечает сообщением TS_UNACCEPTABLE Notify;
- если политика ответчика позволяет принимать весь трафик, покрываемый селекторами TSi и TSr, сужения диапазона не требуется и ответчик может вернуть те же значения TSi и TSr;
- если политика ответчика позволяет принимать первый селектор TSi и TSr, ответчик **должен** сузить диапазон предложенных селекторов до подмножества, включающего первый выбор инициатора; в приведенном выше примере ответчик может возвратить TSi с адресами (198.51.100.43 - 198.51.100.43) для всех портов и протоколов IP;
- если политика ответчика не позволяет принимать первый селектор TSi и TSr, ответчик сужает диапазон до приемлемого подмножества TSi и TSr.

[1]Security Policy Database - база данных о правилах безопасности.
[2]Traffic Selector - селектор трафика.
[3]Traffic Selector-initiator - селекторы трафика инициатора.
[4]Traffic Selector-responder - селекторы трафика ответчика.

После сужения диапазона может возникнуть несколько приемлемых подмножеств, объединение которых не приемлемо. В таких случаях ответчик выбирает одно из приемлемых подмножеств и **может** включить в отклик уведомление ADDITIONAL_TS_POSSIBLE, которое информирует о том, что ответчик сузил диапазон полученных селекторов, но приемлемы и другие селекторы в отдельных SA. С этим типом сообщений Notify не связано каких-либо данных. Такие ситуации могут возникать при разной конфигурации инициатора и ответчика. Если обе стороны согласны с гранулярностью туннелей, инициатор никогда не будет запрашивать более широкий туннель, нежели ответчик готов воспринимать.

Вполне возможно, что политика ответчика будет включать множество небольших диапазонов, охватываемых селектором трафика инициатора, но в соответствии с политикой для каждого из этих диапазонов трафик следует передавать через разные SA. В использованном выше примере политика ответчика может требовать организации для каждой пары адресов отдельно согласованной связи Child SA. Если инициатор генерирует запрос не на основе имеющегося для передачи пакета, а (например) при старте, у него не будет достаточно специфичных первых селекторов трафика, помогающих ответчику выбрать корректный диапазон. В этом случае у ответчика не будет возможности определить, какую пару адресов следует включить в этот туннель и он должен будет делать предположение или отвергнет запрос с сообщением SINGLE_PAIR_REQUIRED Notify.

Сообщение об ошибке SINGLE_PAIR_REQUIRED показывает, что запрос CREATE_CHILD_SA не воспринят, поскольку его отправитель желает воспринимать только селекторы трафика, задающие одну пару адресов. Предполагается, что запрашивающая сторона ответит на уведомление запросом SA только для конкретного трафика, который она пытается переслать.

Отдельные реализации имеют правила, требующие организации отдельных SA для каждой пары адресов. В результате этого, если для ответчика приемлема только часть TSi и TSr, предложенных инициатором, ему **следует** сузить диапазон селекторов до приемлемого подмножества, а не использовать SINGLE_PAIR_REQUIRED.

2.9.1. Селекторы трафика, нарушающие свою политику

При создании новой SA инициатор должен избегать предложения селекторов трафика, нарушающих его собственную политику. При нарушении этого правила может отбрасываться допустимый трафик. Если используются декоррелированные правила из [IPSECARCH], таких нарушений не может возникать.

Это лучше всего проиллюстрировать на примере. Предположим, что хост A имеет политику, в соответствии с которой трафик по адресу 198.51.100.66 передается через хост B с использованием шифрования AES, а трафик для остальных адресов сети 198.51.100.0/24 также передается через хост B, но с шифрованием 3DES. Предположим также, что B воспринимает любые комбинации AES и 3DES.

Если хост A предлагает SA, которая использует 3DES и включает TSr с блоком адресов (198.51.100.0-198.51.100.255), это будет приемлемо для хоста B. Хост B может также использовать эту SA для передачи трафика с адреса 198.51.100.66, но такие пакеты будут отбрасываться хостом A, поскольку он требует использования для них шифрования AES. Даже если хост A создаст новую SA только для 198.51.100.66 (использующего AES), хост B может свободно продолжать использование первой SA для этого трафика. В такой ситуации, предлагая SA хосту A, следует соблюдать свою политику и предлагать TSr, содержащий ((198.51.100.0-198.51.100.65),(198.51.100.67-198.51.100.255)) вместо единого блока.

В общем случае, если (1) инициатор делает предложение «для трафика X (TSi/TSr), создать SA», (2) для некоторого подмножества X' инициатор на деле не будет принимать X' через SA и (3) инициатор желает принимать трафик X' через другую связь SA' (!=SA), корректный трафик может необоснованно отбрасываться, поскольку ответчик может использовать для трафика X' как SA, так и SA' .

2.9.2. Селекторы трафика при смене ключей

Смена ключей используется для замены существующей Child SA на новую защищенную связь. Если новая SA будет позволять сужение набора селекторов по сравнению с исходной, разрешенный в прежней SA трафик будет отбрасываться в новой, нарушая смысл «замены». По этой причине в новой SA **недопустимо** задавать суженный по сравнению с оригиналом набор селекторов. Если новой SA требуется более узкий набор селекторов по сравнению с действующей, это означате, что политика была изменена там, что текущая SA не соответствует ей. В таком случае SA уже должна быть удалена после вступления в силу измененной политики.

Когда инициатор пытается сменить ключи для Child SA, ему **следует** предлагать такие же селекторы трафика, которые используются в существующей Child SA, или включающее старые селекторы более широкое множество. Т. е., селекторы будут совпадать или включать в себя действующие селекторы активной (декоррелированной) политики. Ответчику **недопустимо** сужать селекторы трафика по сравнению с используемыми в настоящее время.

Поскольку SA после смены ключей никогда не может иметь более узкий по сравнению с действующим набор селекторов, селекторы из пакета не требуются и передавать их **не следует**.

2.10. Nonce

Каждое сообщение IKE_SA_INIT содержит значение nonce, которое используется в качестве аргумента криптографической функции. Запросы CREATE_CHILD_SA и отклики CREATE_CHILD_SA также содержат nonce, которые служат для добавления «свежести» методам создания ключей, используемым при генерации ключей для Child SA, а также для обеспечения создания действительно псевдослучайных битов из ключа Diffie-Hellman. Значения nonce, используемые в IKEv2, **должны** выбираться случайным образом, **должны** быть размером не менее 128 битов, а также **должны** быть по размеру не менее половины размера результата согласованной псевдослучайной функции (PRF). Однако инициатор выбирает nonce до того, как узнает результат согласования. По этой причине размер nonce должен быть достаточно велик для всех предлагаемых функция PRF. Если для ключей и nonce используется общий источник случайных чисел, следует принимать меры против компрометации ключей при использовании nonce.

2.11. Адреса и номера портов

IKE работает по протоколу UDP через порты 500 и 4500, неявно устанавливая связь ESP и AH с теми же адресами IP, которые используются IKE. Однако адреса и номера портов во внешних заголовках не защищаются криптографически, а протокол IKE рассчитан также на работу через системы трансляции адресов (NAT[1]). Реализации **должны** воспринимать входящие запросы, если номер порта источника отличается от 500 и 4500, а также **должны** отвечать по адресам и номерам портов, с которых были получены запросы. Реализации **должны** указывать адрес и номер порта, по которым были получены запросы, в полях адреса и номера порта отправителя передаваемых откликов. Функции IKE идентичны для протоколов IPv4 и IPv6.

2.12. Многократное использование значений Diffie-Hellman

IKE генерирует ключевой материал, используя эфемерный обмен Diffie-Hellman для достижения преимуществ «совершенной защиты[2]». Это означает, что после разрыва соединения и забывания соответствующих ключей даже тот, кто сумел записать все данные из соединения и получить доступ ко всем долгосрочным ключам обеих конечных точек, не сможет восстановить ключи, использованные для защиты соединения без перебора всех ключей в пространстве ключей сеанса (brute force search).

Для достижения совершенной защиты каждая из конечных точек по завершении соединения **должна** забыть не только использованные в нем ключи, но и любую информацию, которая могла бы использоваться для восстановления этих ключей.

Поскольку расчет значений Diffie-Hellman требует значительных вычислительных ресурсов, конечная точка может принять решение о неоднократном использовании этих значений DH при организации последующих соединений. Для таких действий имеется несколько разумных стратегий. Конечная точка может менять значения DH периодически, хотя это приведет к некоторому снижению уровня защиты, если срок жизни соединения окажется меньше периода смены значений DH. Можно также отслеживать использование значений DH в каждом соединении и удалять информацию, связанную со значением DH, только после закрытия соответствующего соединения. Это позволит многократно использовать значения DH без снижения уровня защиты но с некоторым ростом издержек на поддержку большего числа состояний.

Решение о многократном использовании значений Diffie-Hellman и режиме такого использования является частным в том смысле, что оно не оказывает влияния на интероперабельность. Реализация, использующая значения DH неоднократно, **может** запоминать значения DH, использованные другой точкой в прошлых обменах, и при повторном их появлении избежать второй половины расчета. Анализ защиты для этого случая и дополнительное рассмотрение вопросов безопасности при многократном использовании эфемерных ключей Diffie-Hellman приведены в работах [REUSE] и [RFC6989].

2.13. Генерация ключевого материала

В контексте IKE SA согласуются четыре криптографических алгоритма - алгоритмы шифрования и защиты целостности, группа Diffie-Hellman и псевдослучайная функция (PRF). Функция PRF служит для подготовки ключевого материала, используемого всеми криптоалгоритмами IKE SA и Child SA.

Предполагается, что алгоритмы шифрования и защиты целостности используют ключи фиксированного размера и в качестве ключа может служить произвольное случайно выбранное значение этого размера. Для алгоритмов, способных работать с ключами разного размера, **должен** задаваться фиксированный размер ключа при согласовании криптографических преобразований (см. определение атрибута преобразования Key Length в параграфе 3.3.5). Для алгоритмов, которые принимают не все значения ключей (таких, как DES или 3DES с четностью ключей), алгоритм выбора ключей из набора произвольных значений **должен** задаваться криптографическим преобразованием. Для функций защиты целостности на базе хэшированных кодов аутентификации сообщений (HMAC[3]) фиксированным размером ключа является размер вывода используемой хэш-функции.

Предполагается, что функции PRF воспринимают ключи любого размера, но имеют предпочтительный размер ключей. Этот размер **должен** использоваться в качестве размера SK_d, SK_pi и SK_pr (см. параграф 2.14). Для функций PRF на базе конструкций HMAC предпочтительный размер ключей равен размеру результата используемой хэш-функции. Остальные типы PRF **должны** задавать предпочтительный размер ключей.

Ключевой материал всегда берется из вывода согласованного алгоритма PRF. Поскольку количество требуемого ключевого материала может превышать размер вывода PRF, функция PRF используется в режиме итераций. Обозначение prf+ указывает на функцию, результатом которой является псевдослучайный поток, основанный на вводе данных в случайную функцию prf.

В приведенных ниже примерах | обозначает конкатенацию. Функция prf+ определяется, как

```
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)
...
```

Этот процесс продолжается, пока на выходе функции prf+ не будет получен полный объем ключевого материала, требуемого для расчета всех нужных ключей. Ключи берутся из результата функции без учета границ (т. е, если требуется 256-битовый ключ AES[4] и 160-битовый ключ HMAC, а функция prf генерирует на выход 160 битов, ключ AES будет состоять из T1 и начала T2, а ключ HMAC из остатка T2 и начала T3).

[1]Network Address Translation.
[2]Perfect forward secrecy.
[3]Hashed Message Authentication Code.
[4]Advanced Encryption Standard.

Константа, добавляемая после каждого результата prf, представляет собой один октет. Функцию prf+ не используют для увеличения размера вывода prf более, чем в 255 раз.

2.14. Генерация ключевого материала для IKE SA

Общие (разделяемые) ключи генерируются, как описано ниже. Значение SKEYSEED рассчитывается из значений nonce, передаваемых в процессе обмена IKE_SA_INIT и общего секрета Diffie-Hellman, организованного в процессе этого обмена. Число SKEYSEED используется для расчета семи других секретов - SK_d применяется при генерации новых ключей для Child SA, организуемых в рамках IKE SA, SK_ai и SK_ar служат в качестве ключей алгоритма защиты целостности для аутентификации сообщений при последующих обменах, SK_ei и SK_er применяются для шифрования (и расшифровки) при всех последующих обменах, SK_pi и SK_pr используются при генерации данных AUTH. Размеры SK_d, SK_pi и SK_pr **должны** совпадать с предпочтительным размером ключей согласованной ранее функции PRF.

SKEYSEED и производные от него значения 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 том порядке, в котором создается выход prf+). g^{ir} является общим секретом из эфемерного обмена Diffie-Hellman. Значение g^ir представляется, как строка октетов в порядке big endian и дополняется при необходимости нулями до требуемого размера. Ni и Nr are представляют собой одноразовые значения nonce, вырезаемые из любых заголовков. В силу исторических причин (для совместимости со старыми версиями) имеется две функции PRF, которые в таком расчете имеют специфическую трактовку. Если согласованной PRF является AES-XCBC-PRF-128 [AESXCBCPRF128] или AES-CMAC-PRF-128 [AESCMACPRF128], только первые 64 бита Ni и первые 64 бита Nr используются при расчете SKEYSEED, а все остальные биты служат для ввода в prf+.

Разные направления потока трафика используют различные ключи. Ключами, служащими для защиты трафика от исходного инициатора являются SK_ai и SK_ei, а ключами для другого направления - SK_ar и SK_er.

2.15. Аутентификация IKE SA

Когда расширяемая аутентификация (см. параграф 2.16) не используется, партнеры аутентифицируются с помощью цифровых подписей (или MAC с использованием дополненного общего секрета, как описано ниже в этом параграфе) для блоков данных. В этих расчетах IDi' и IDr' - все данные ID за исключением фиксированного заголовка. Для ответчика подписываемые октеты начинаются с первого октета первого значения SPI в заголовке второго сообщения (отклик IKE_SA_INIT) и заканчивается последним октетом последнего элемента данных во втором сообщении. К этому в конце добавляется (для расчета цифровой подписи) значение nonce инициатора Ni (просто значение, а не содержащий его элемент данных) и prf(SK_pr, IDr'). Отметим, что ни nonce Ni, ни значение prf(SK_pr, Idr'), сами по себе, не передаются. Подобно этому, инициатор подписывает первое сообщение (запрос IKE_SA_INIT), начиная с первого октета первого SPI в заголовке и заканчивая последним октетом последнего элемента данных. В конце этого добавляется (для расчета цифровой подписи) значение nonce ответчика Nr и значение prf(SK_pi, IDi'). Для защиты крайне важно использование в подписи каждой стороны значения nonce другой стороны.

Подписанные инициатором октеты можно описать следующим образом:

```
InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
GenIKEHDR = [ 4 октета со значением 0, если применяется порт 4500 ] | RealIKEHDR
RealIKEHDR =  SPIi | SPIr |  . . . | Length
RealMessage1 = RealIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)
```

Подписанные ответчиком октеты можно описать, как:

```
ResponderSignedOctets = RealMessage2 | NonceIData | MACedIDForR
GenIKEHDR = [ 4 октета со значением 0, если применяется порт 4500 ] | RealIKEHDR
RealIKEHDR =  SPIi | SPIr |  . . . | Length
RealMessage2 = RealIKEHDR | RestOfMessage2
NonceIPayload = PayloadHeader | NonceIData
ResponderIDPayload = PayloadHeader | RestOfRespIDPayload
RestOfRespIDPayload = IDType | RESERVED | RespIDData
MACedIDForR = prf(SK_pr, RestOfRespIDPayload)
```

Отметим, что в расчет подписи включаются все элементы данных (payload), в том числе и те, которые не определены в данном документе. Если первое сообщение в обмене (например, cookie ответчика и/или другая группа Diffie-Hellman) повторяется несколько раз, подписывается последняя версия сообщения.

Опционально сообщения 3 и 4 **могут** включать сертификат или цепочку сертификатов, подтверждающих, что ключ, использованный для создания цифровой подписи, относится к имени, указанному в элементе данных ID. Подпись или код MAC будут рассчитываться с использованием алгоритма, диктуемого типом ключа, применяемого подписавшим и указанного в поле Auth Method элемента данных Authentication. Инициатор и ответчик не обязаны использовать для подписи один криптографический алгоритм. Выбор алгоритма зависит от типа имеющихся ключей. В частности, инициатор может использовать общий ключ, а ответчик иметь открытый ключ подписи и сертификат. На практике зачастую (но не обязательно) при использовании общего секрета для аутентификации в обоих направлениях применяется один ключ.

Отметим, что обычной, но зачастую небезопасной практикой является использование общего ключа, полученного из выбранного пользователем пароля без включения иных источников случайных данных. Недостаточная безопасность такого подхода обусловлена тем, что выбираемые пользователями пароли не обеспечивают непредсказуемости, достаточной для предотвращения атак по словарю (приложениям, использующим парольную аутентификацию для загрузки и IKE SA, следует выбирать метод аутентификации из параграфа 2.16, который предотвращает атаки по

словарю в режиме off-line). Заранее созданные общие ключи должны обеспечивать уровень непредсказуемости, достаточный для самого сильного из согласуемых ключей. В случае заранее созданного общего ключа (pre-shared key), значение AUTH рассчитывается, как показано ниже.

Для инициатора

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

Для ответчика

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

где строка «Key Pad for IKEv2[1]» представляет собой 17 символов ASCII без завершающего null-символа. Общий секрет может иметь переменный размер. Строка заполнения добавляется так, чтобы при создании общего секрета из пароля реализации IKE не требовалось сохранять пароля в открытом виде, а можно было бы сохранить значение prf(Shared Secret,"Key Pad for IKEv2"), которое не может использоваться в качестве эквивалента пароля за пределами IKEv2. Как было отмечено выше, создание общего секрета из пароля не вполне безопасно. Такая конструкция используется в предположении, что люди все равно будут делать это. Интерфейс управления, посредством которого обеспечивается общий секрет, **должен** воспринимать строки ASCII размером, по крайней мере 64 октета, добавление null-символа завершения перед использованием разделяемого секрета **недопустимо**. Интерфейс управления **может** воспринимать другие кодировки символов, если задан алгоритм для представления символов в форме двоичной строки.

Имеется два типа аутентификации EAP[2] (см. параграф 2.16), каждый из которых использует разные значения в описанных выше расчетах AUTH. Если метод EAP служит для генерации ключей, главный сеансовый ключ (MSK[3]) при расчете меняется на общий секрет. Для методов, не используемых для генерации ключей, SK_pi и SK_pr заменяются общим секретом в обоих расчетах AUTH.

2.16. Методы EAP

В дополнение к аутентификации с использованием подписей на основе открытых ключей и общих секретов, IKE поддерживает аутентификацию на основе методов, определенных в RFC 3748 [EAP]. Обычно эти методы являются асимметричными (предназначены для аутентификации пользователей на сервере) и могут не быть взаимными. По этой причине такие протоколы обычно используются инициатором для аутентификации самого себя перед ответчиком и **должны** применяться совместно с аутентификацией на основе подписи с открытым ключом для представления ответчика инициатору. Эти методы часто связывают с механизмами, которые называют «унаследованной аутентификацией[4]».

Хотя в этом документе [EAP] упоминается с учетом добавления в будущем новых методов без обновления данной спецификации, некоторые простые вариации здесь рассматриваются. [EAP] определяет протокол аутентификации, требующий переменного числа сообщений. Расширяемая аутентификация реализуется в IKE в форме дополнительных обменов IKE_AUTH, которые **должны** быть завершены для инициализации IKE SA.

```
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 (успех)}
HDR, SK {AUTH}                  -->
                                <--  HDR, SK {AUTH, SAr2, TSi, TSr }
```

Инициатор показывает желание использовать EAP, опуская поле AUTH в первом сообщении обмена IKE_AUTH (отметим, что это поле требуется для других вариантов аутентификации, поэтому в данном документе не отмечено, как опциональное). Путем включения элемента IDi и исключения AUTH инициатор заявляет свою подлинность, но не подтверждает ее. Если ответчик согласен использовать метод EAP, он помещает элемент EAP в отклик обмена IKE_AUTH и откладывает передачу SAr2, TSi и TSr, пока аутентификация инициатора не будет завершена в следующем обмене IKE_AUTH. Для случая минимального метода EAP организация начальной SA показана на рисунке справа.

Как указано в параграфе 2.2, при использовании EAP каждая пара изначальных сообщений при организации IKE SA будет использовать увеличивающиеся порядковые номера - сообщения IKE_AUTH первой пары будут иметь номер 1, второй - 2 и т. д.

Для методов EAP, создающих общий ключ в качестве побочного эффекта аутентификации, этот ключ **должен** использоваться как инициатором, так и ответчиком для генерации данных AUTH в сообщениях 7 и 8 с использованием синтаксиса общих секретов, описанного в параграфе 2.15. Общий ключ от EAP представляет собой поле, которое в спецификации EAP названо MSK. Этот общий ключ генерируется в процессе обмена IKE и его **недопустимо** использовать для каких-либо целей.

Методы EAP, которые не организуют общего ключа, **не следует** применять, поскольку они могут быть подвержены MITM-атакам [EAPMITM], если эти методы EAP используются в других протоколах, не использующих аутентифицируемого сервером туннеля. Более подробное обсуждение этой ситуации приводится в разделе «Вопросы безопасности». При использовании методов EAP, не генерирующих общего ключа, элементы AUTH в сообщениях 7 и 8 **должны** генерироваться с использованием SK_pi и SK_pr, соответственно.

От инициатора IKE SA, использующего EAP, требуется возможность расширения начального протокольного обмена по крайней мере до 10 обменов IKE_AUTH в тех случаях, когда ответчик передает уведомления и/или отклики на приглашение аутентификации. После успешного завершения протокольного обмена, определенного выбранным методом аутентификации EAP, ответчик **должен** передать элемент EAP, содержащий сообщение Success. При отказе

[1]Key Pad for IKEv2 - заполнение ключа для IKEv2.
[2]Extensible Authentication Protocol - расширяемый протокол аутентификации.
[3]Master session key.
[4]Legacy Authentication.

ответчик **должен** передать элемент EAP, содержащий сообщение Failure. Ответчик **может** в любой момент прервать обмен IKE, передав элемент EAP с сообщением Failure.

После расширенного обмена элементы EAP AUTH **должны** быть включены в два сообщения после сообщения с EAP Success.

При использовании инициатором аутентификации EAP возможны ситуации, когда содержимое IDi используется только для целей AAA[1] при маршрутизации и выбора применяемого метода EAP. Это значение может отличаться от отождествлений, аутентифицированных методом EAP. Важно, чтобы при просмотре правил и принятии решения о предоставлении доступа использовались актуальные аутентифицированные отождествления. Зачастую сервер EAP реализуется на отдельном сервере AAA, который обменивается данными с ответчиком IKEv2. В таких случаях, аутентифицированное отождествление (если оно отличается от данных IDi) будет передаваться от сервера AAA ответчику IKEv2.

2.17. Генерация ключевого материала для Child SA

Одна дочерняя связь Child SA создается обменом IKE_AUTH, а дополнительные Child SA могут создаваться в обменах CREATE_CHILD_SA. Ключевой материал для них генерируется следующим образом

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

Здесь Ni и Nr - значения nonce из обмена IKE_SA_INIT, если это запрос на создание первой Child SA или свежие значения Ni и Nr из обмена CREATE_CHILD_SA для последующих дочерних связей.

Для CREATE_CHILD_SA, включающих обмен Diffie-Hellman, ключевой материал генерируется по формуле

$$KEYMAT = prf+(SK\_d, g^{ir}\ (new) \mid Ni \mid Nr)$$

где g^{ir} (new) - общий секрет из эфемерного обмена Diffie-Hellman в данном CREATE_CHILD_SA (представляется, как строка октетов с порядком big endian, дополненная при необходимости нулями в старших битах для получения размера модуля).

Одно согласование CREATE_CHILD_SA может приводить к организации множества защищенных связей. Связи ESP и AH существуют попарно (по одной для каждого направления), поэтому для этих протоколов создается две SA в одном согласовании Child SA. Кроме того, согласование Child SA может включать некоторые будущие протоколы IPsec в дополнение (или взамен) к ESP или AH (например, ROHC_INTEG [ROHCV2]). В любом случае ключевой материал для каждой Child SA **должен** браться из расширенного KEYMAT с использованием приведенных ниже правил.

- Все ключи для SA, передающих данные от инициатора к ответчику, берутся до ключей для SA обратного направления.
- Если согласовано множество протоколов IPsec, ключевой материал для каждой Child SA берется в порядке следования протокольных заголовков в инкапсулированном пакете.
- Если протокол IPsec требует множества ключей, порядок их извлечения из ключевого материала SA требуется описать в спецификации протокола. Для ESP и AH в [IPSECARCH] определен следующий порядок - ключ шифрования (если он есть) **должен** браться из первых битов, а ключ защиты целостности (если он есть) - из оставшихся.

Каждый криптографический алгоритм берет фиксированное число битов ключевого материала, заданное в спецификации или согласованное в данных SA (см описание размеров ключей в параграфе 2.13 и определение алгоритма преобразования Key Length в параграфе 3.3.5).

2.18. Смена ключей IKE SA с помощью обмена CREATE_CHILD_SA

Обмен CREATE_CHILD_SA можно использовать для смены ключей существующей IKE SA (см. параграфы 1.3.2 и 2.8). Новые значения SPI инициатора и ответчика поставляются в полях SPI субструктур Proposal внутри элементов SA (не поля SPI в заголовке IKE). Элементы TS опускаются при смене ключей IKE SA. Значение SKEYSEED для новой IKE SA рассчитывается с использованием SK_d из существующей IKE SA следующим образом:

$$SKEYSEED = prf(SK\_d\ (old), g^{ir}\ (new) \mid Ni \mid Nr)$$

где g^{ir} (new) - общий секрет из эфемерного обмена Diffie-Hellman в данном CREATE_CHILD_SA (представляется, как строка октетов с порядком big endian, дополненная при необходимости нулями в старших битах для получения размера модуля), а Ni и Nr - значения nonce из любых заголовков.

Старая и новая IKE SA могут иметь разные функции PRF. Поскольку обмен, связанный со сменой ключей, происходит в старой IKE SA, для генерации SKEYSEED используется функция PRF из старой IKE SA.

Основной причиной смены ключей IKE SA является предотвращение возможности использования скомпрометированного старого ключевого материала для получения информации о текущих ключах и наоборот. Следовательно, реализации **должны** выполнять новый обмен Diffie-Hellman при смене ключей IKE SA. Иными словами, для инициатора **недопустимо** предлагать значение NONE в качестве преобразования Diffie-Hellman, а для ответчика **недопустимо** принимать такие предложения. Это означает, что для успешного обмена с заменой ключей IKE SA всегда используются элементы данных KEi/KEr.

В новой IKE SA значения счетчиков сообщений **должны** сбрасываться в 0.

SK_d, SK_ai, SK_ar, SK_ei и SK_er рассчитываются из SKEYSEED, как описано в параграфе 2.14, с использованием SPIi, SPIr, Ni и Nr из нового обмена и функции PRF из новой IKE SA.

2.19. Запрос внутреннего адреса из удаленной сети

Наиболее часто встречающимся сценарием взаимодействия между конечной точкой и защитным шлюзом является необходимость получения конечной точкой IP-адреса из сети, защищенной шлюзом (возможно, динамического). Запрос на получение такого адреса может быть включен в любой запрос на создание Child SA (включая неявный запрос в

[1]Authentication, Authorization, and Accounting — аутентификация, проверка полномочий и учет.

сообщении 3) с использованием элемента CP. Отметим, однако, что обычно выделяется только один адрес IP в процессе обмена IKE_AUTH. Этот адрес сохраняется по крайней мере до удаления IKE SA.

```
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}
```

Эта функция обеспечивает выделение адреса для удаленного клиента IPsec (IRAC[1]), пытающегося организовать туннель в сеть, защищенную сервером удаленного доступа IPsec (IRAS[2]). Поскольку обмен IKE_AUTH создает IKE SA и Child SA, клиент IRAC **должен** запросить контролируемый сервером IRAS адрес (и, возможно, другую информацию, относящуюся к защищенной сети) в обмене IKE_AUTH. Сервер IRAS может получать адрес для IRAC из любого числа источников типа серверов DHCP/BOOTP[3] (протокол загрузки) или выделять его из своего пула.

Во всех случаях элемент данных CP **должен** помещаться до элемента SA. В вариантах протокола с множественными обменами IKE_AUTH элементы CP **должны** помещаться в сообщения, содержащие элементы SA.

Вызов CP(CFG_REQUEST) **должен** включать по крайней мере атрибут INTERNAL_ADDRESS (IPv4 или Ipv6), но **может** содержать любое число дополнительных атрибутов, которые инициатор хочет вернуть в отклике.

Пример сообщения от инициатора к ответчику может иметь вид:

```
CP(CFG_REQUEST)= INTERNAL_ADDRESS()
TSi[4] = (0, 0-65535,0.0.0.0-255.255.255.255)
TSr = (0, 0-65535,0.0.0.0-255.255.255.255)
```

Сообщение от ответчика инициатору:

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

Все возвращаемые значения зависят от реализации. Как можно видеть в приведенных примерах, сервер IRAS **может** также передать другие атрибуты, которые не были включены в CP(CFG_REQUEST) и **может** игнорировать любые необязательные атрибуты, которые он не поддерживает.

Ответчику **недопустимо** передавать CFG_REPLY без получения сначала запроса CP(CFG_REQUEST) от инициатора, поскольку мы не хотим, чтобы сервер IRAS выполнял ненужные просмотры конфигурации в случаях, когда IRAC не может обработать REPLY.

В случаях, когда конфигурация IRAS требует, чтобы элемент CP использовался для данного отождествления IDi, но IRAC не смог передать CP(CFG_REQUEST), сервер IRAS **должен** отказаться от выполнения запроса и прервать создание Child SA с помощью сообщения об ошибке FAILED_CP_REQUIRED. Это сообщение не относится к критическим для IKE SA и просто приводит к отказу при создании Child SA. Инициатор может исправить ситуацию с помощью нового запроса с элементом Configuration. С сообщением FAILED_CP_REQUIRED не связано каких-либо данных.

2.20. Запрос версии партнера

Узел IKE, желающий узнать версию IKE своего партнера, **может** использовать описанный ниже метод. Это пример конфигурационного запроса в обмене INFORMATIONAL после организации IKE SA и первой Child SA.

```
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.")
```

Реализация IKE **может** отвергнуть запрос на предоставление сведений о версии до аутентификации и даже после ее завершения в том случае, когда известно о тех или иных недостатках в защите. В таких случаях **должна** возвращаться пустая строка или сообщение без элемента CP, если CP не поддерживается.

2.21. Обработка ошибок

При обработке IKE может возникать множество различных ошибок. Общим правилом является передача в ответ на запросы с некорректным форматом или неприемлемые по соображениям политики (например, несоответствие криптографических алгоритмов) отклика с элементами Notify, указывающими на ошибку. Решение о передаче такого отклика зависит от того, была ли аутентифицирована связь IKE SA.

Если ошибка возникает при разборе или обработке отклика, в соответствии с общим правилом не следует передавать в ответ сообщение об ошибке, поскольку отклики не должны вызывать генерации новых запросов (а новый запрос является единственным способом передать ответчику сообщение об ошибке). В случае ошибки при разборе или обработке отклика его получателю следует сбросить состояние IKE (например, передать Delete для SA).

[1]IPsec Remote Access Client.
[2]IPsec Remote Access Server.
[3]Bootstrap Protocol.
[4]Селекторы трафика содержат протокол, а также диапазоны портов и адресов.

Только отказы при аутентификации (AUTHENTICATION_FAILED и отказ EAP) и сообщения с некорректным форматом (INVALID_SYNTAX) ведут к удалению IKE SA без необходимости использования явного обмена INFORMATIONAL с элементом Delete. Другие ошибки **могут** требовать такого обмена в соответствии с политикой. Если обмен прерывается отказом EAP Failure, уведомление AUTHENTICATION_FAILED не передается.

2.21.1. Обработка ошибок в IKE_SA_INIT

Ошибки, возникающие до завершения организации криптографически защищенной IKE SA, следует обрабатывать очень осторожно. Здесь требуется найти компромисс между желанием помочь партнеру в диагностике проблем (и, таким образом, избавиться от ошибки) и возможностью оказаться вовлеченным в DoS-атаку с использованием поддельных пакетов.

При обменах IKE_SA_INIT любое уведомление об ошибке вызывает отказ. Отметим, что некоторые уведомления (например, COOKIE, INVALID_KE_PAYLOAD или INVALID_MAJOR_VERSION) позволяют впоследствии завершить обмен успешно. Поскольку все уведомления об ошибках полностью не аутентифицированы, получателю следует продолжать попытки в течение некоторого времени, прежде, чем дать отказ. Получателю не следует незамедлительно выполнять какие-либо действия на основе полученных уведомлений об ошибке, если в этой спецификации не определено соответствующих корректировочных мер, как для уведомлений COOKIE, INVALID_KE_PAYLOAD и INVALID_MAJOR_VERSION.

2.21.2. Обработка ошибок в IKE_AUTH

При возникновении любой ошибки в обмене IKE_AUTH, вызывающей отказ аутентификации независимо от причины (некорректный общий секрет или идентификатор, недоверенный эмитент сертификата, отозванный или просроченный сертификат и т. п.), **следует** передавать уведомление AUTHENTICATION_FAILED. Если ошибка возникла на стороне ответчика, уведомление возвращается в защищенном отклике и обычно не содержит других элементов. Хотя сообщения IKE_AUTH шифруются и целостность их защищается, если получивший такое сообщение партнер еще не аутентифицирован на другой стороне, ему следует с осторожностью воспринимать информацию об ошибке.

При возникновении ошибки на стороне инициатора уведомление **может** быть передано в отдельном обмене INFORMATIONAL, который обычно не включает других элементов. Это является исключением из общего правила не начинать новых обменов результате ошибок в откликах.

Однако следует отметить, что запросы с неподдерживаемыми критичными элементами или некорректно форматированные сообщения (а не просто с некорректными полями данных) **должны** отвергаться целиком и **должны** вызывать только отклик в форме уведомления UNSUPPORTED_CRITICAL_PAYLOAD или INVALID_SYNTAX. Получателю не следует проводить в таких случаях верификацию данных, относящихся к аутентификации.

Если проверка подлинности успешно завершилась в обмене IKE_AUTH, это означает, что связь IKE SA организована, однако при организации Child SA или запросе конфигурационной информации может возникнуть отказ. Такие отказы не ведут к автоматическому удалению IKE SA. В частности, ответчик может включить все элементы, связанные с аутентификацией (IDr, CERT, AUTH) при отправке уведомления об ошибке для дополняемых обменов (FAILED_CP_REQUIRED, NO_PROPOSAL_CHOSEN и т. п.), а инициатору **недопустимо** на основании этого отказываться от аутентификации. Инициатор **может**, конечно, по соображениям политики позднее удалить эту IKE SA.

В обмене IKE_AUTH или непосредственно следующим за ним обмене INFORMATIONAL (в случае ошибок при обработке отклика на IKE_AUTH) только уведомления UNSUPPORTED_CRITICAL_PAYLOAD, INVALID_SYNTAX и AUTHENTICATION_FAILED вызывают удаление или отказ от создания IKE SA без использования элемента Delete. Документы расширений могут определять дополнительные уведомления об ошибках, использующие такую семантику, но **недопустимо** использовать их, пока партнер не указал, что способен их понимать (например, используя данные Vendor ID).

2.21.3. Обработка ошибок после аутентификации IKE SA

После аутентификации IKE SA все запросы с ошибками **должны** вызывать передачу откликов с уведомлениями об ошибках.

В нормальных ситуациях не возникает случаев, когда нормальный отклик одного из партнеров вызывает ошибку у другого партнера, следовательно, не должно возникать причин передавать на другую сторону сообщения иначе, чем в откликах. Поскольку передача таких сообщений в форме обмена INFORMATIONAL может вызывать дополнительные ошибки, приводящие к петлям сообщений, такие сообщения передавать **не следует**. Если представляется, что ошибка указывает на то, что партнеры находятся в разных состояниях, может оказаться разумным удаление IKE SA, сброс состояния и организация нового соединения.

Если разбирающий запрос партнер видит, что запрос имеет некорректный формат (после проверки кода аутентификации сообщения и окна), и возвращает уведомление INVALID_SYNTAX, такое уведомление рассматривается обеими сторонами, как критическое и требующее удаления IKE SA без необходимости явного включения элемента Delete.

2.21.4. Обработка ошибок за пределами IKE SA

Узлы должны ограничивать скорость, с которой могут передаваться сообщения в ответ на незащищенные 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 организацию новой IKE SA), это может быть результатом недавней аварии на узле. Если сообщение помечено, как отклик, узел может проверить подозрительное событие, но отвечать на сообщение **недопустимо**. Если сообщение помечено, как запрос, узел может проверить подозрительное событие и **может** передать отклик на сообщение. Если отклик передается, он **должен** направляться по адресу IP и с номером порта, которые были указаны в полях отправителя полученного пакета с сохранением значений IKE SPI и Message ID из принятого пакета. Отклик **недопустимо** защищать криптографически и он **должен** содержать элемент Notify из INVALID_IKE_SPI. Уведомление INVALID_IKE_SPI показывает, что сообщение IKE было получено для нераспознанного SPI (это обычно говорит о перезагрузке получателя с потерей существовавших IKE SA).

Партнеру, получившему такой незащищенный элемент Notify, **недопустимо** отвечать на сообщение и **недопустимо** менять состояние для любой из существующих SA. Сообщение может быть обманным или являться корректным откликом на переданный отвечающему узлу обманный запрос. Узлу следует трактовать такие сообщения (а также сетевые сообщения типа ICMP destination unreachable), как намек на возможное наличие проблем с SA для этого адреса IP и целесообразность проверки жизнеспособности IKE SA. Реализациям **следует** ограничивать частоту таких проверок, чтобы не оказаться вовлеченным в организацию DoS-атак.

Если ошибка происходит вне контекста запроса IKE (например, узел получает сообщения ESP для несуществующего SPI), узлу **следует** инициировать обмен INFORMATIONAL с элементом Notify, описывающим проблему.

Узлу, получившему подозрительное сообщение с адреса IP (и номера порта, если используется NAT), с которым он имеет IKE SA, **следует** передать элемент IKE Notify в обмене IKE INFORMATIONAL через эту связь SA. Получателю **недопустимо** менять состояние какой-либо SA в результате приема подозрительных сообщений, но можно проверить подозрительные события.

2.22. Компрессия IPComp

Использование компрессии IP [IP-COMP] может быть согласовано в процессе организации Child SA. Хотя сжатие IP включает дополнительный заголовок в каждом пакете и индекс параметров сжатия (CPI[1]), виртуальная «ассоциация сжатия» не выходит на пределы содержащей ее связи ESP или AH. Эти ассоциации исчезают при завершении соответствующей связи ESP или AH. Они явно не указываются в элементах Delete.

Согласование сжатия IP отделено от согласования криптографических параметров, связанных с Child SA. Узел, запрашивающий Child SA, **может** анонсировать поддержку одного или множества алгоритмов компрессии а одном или множестве элементов Notify типа IPCOMP_SUPPORTED. Эти сообщения Notify могут включаться только в сообщение, содержащее элемент SA, согласующий Child SA, и показывает желание отправителя использовать IPComp для этой SA. Отклик **может** показывать приемлемость одного механизма сжатия с помощью элемента Notify типа IPCOMP_SUPPORTED. Такие элементы **недопустимо** помещать в сообщения, не содержащие полей элементов SA.

Данные, связанные с этим сообщением Notify, включают два октета IPComp CPI, за которыми следует один октет Transform ID, после которого могут включаться атрибуты, размер и формат которых определяется значением Transform ID. Сообщение, предлагающее SA, может содержать множество уведомлений IPCOMP_SUPPORTED для указания разных поддерживаемых алгоритмов. В сообщении, принимающем SA, может указываться не более одного алгоритма.

Значения Transform ID приведены в таблице (по состоянию на момент публикации RFC 4306). С тех пор могли появиться новые преобразования, возможно их добавление и после публикации данного документа. Для получения актуальной информации следует обратиться к [IKEV2IANA].

Имя	Номер	Документ
IPCOMP_OUI	1	
IPCOMP_DEFLATE	2	RFC 2394
IPCOMP_LZS	3	RFC 2395
IPCOMP_LZJH	4	RFC 3051

Хотя здесь обсуждалась возможность восприятия множества алгоритмов компрессии и применения разных алгоритмов для каждого направления Child SA, реализациям данной спецификации **недопустимо** воспринимать алгоритм IPComp, который не был предложен, **недопустимо** воспринимать более одного алгоритма, а также **недопустимо** сжимать данные с использованием алгоритма, отличающегося от предложенных и воспринятых при организации данной Child SA.

Побочным эффектом раздельного согласования IPComp и криптографических параметров является невозможность предложить заданные комбинации шифронаборов и IP Compression.

В некоторых случаях компрессия заголовков ROHC[2] может оказаться более подходящей, нежели IP Compression. Документ [ROHCV2] определяет использование компрессии ROHC с IKEv2 и IPsec.

2.23. Работа через NAT

Шлюзы с трансляцией адресов (NAT[3]) являются спорным вопросом. В этом параграфе кратко описано, что и как такие шлюзы делают с трафиком IKE. Многие считают NAT злом и полагают, что не следует разрабатывать протоколы так, чтобы трансляторы работали лучше. IKEv2 задает некоторые не вполне очевидные правила обработки для того, чтобы улучшить работу через NAT.

Существование NAT обусловлено в основном нехваткой адресов IPv4, хотя есть и другие причины. Узлы IP, находящиеся за NAT, могут не иметь уникальных в глобальном масштабе адресов IP, а использовать адреса из неких блоков, обеспечивающих уникальность лишь в пределах расположенной за NAT сети, которые могут одновременно применяться в других сетях, расположенных за другими устройствами NAT. В общем случае узлы, расположенные за NAT, могут взаимодействовать с узлами, расположенными за тем же NAT, и с узлами, имеющими уникальные в глобальном масштабе адреса, но не с узлами, расположенными за другими трансляторами NAT. Из этого правила существуют исключения. Когда расположенные за NAT узлы соединяются с другими узлами в сети Internet, шлюз NAT транслирует IP-адреса отправителей, преобразуя их в некий адрес (адреса), которые будут маршрутизироваться обратно на этот шлюз. Приходящие на шлюз пакеты из сети Internet будут снова транслироваться с заменой адреса получателя на внутренний адрес, чтобы пакет был доставлен нужному узлу за шлюзом.

Системы NAT устроены так, чтобы они оставались «прозрачными» для конечных узлов. Ни программы на узлах за NAT, ни узлы Internet не требуется как-то изменять для работы через NAT. Обеспечение такой прозрачности для некоторых протоколов сложнее, чем для других. Протоколы, включающие в поля данных пакетов IP-адреса конечных точек, не будут работать, пока шлюзы NAT не станут понимать эти протоколы и изменять внутренние поля пакетов, как это делается при трансляции полей заголовков. Однако такое решение по своей природе ненадежно, нарушает требования сетевого уровня и зачастую приводит к возникновению серьезных проблем.

Организация соединений IPsec через NAT вносит свои проблемы. Если соединение организуется в транспортном режиме, замена адресов IP в пакетах будет приводить к отказам при проверке контрольной суммы, а NAT не может ее

[1]Compression parameter index.
[2]Robust Header Compression - отказоустойчивое сжатие заголовков.
[3]Network Address Translation.

исправить, поскольку значение контрольной суммы защищено криптографически. Даже в туннельном режиме возникают проблемы с маршрутизацией, поскольку прозрачная трансляция адресов в пакетах AH и ESP требует реализации в NAT специальной логики, которая эвристична и ненадежна по своей природе. По этим причинам в IKEv2 используется инкапсуляция пакетов IKE и ESP в UDP. Такое кодирование менее эффективно, но существенно упрощает обработку в NAT. В дополнение к этому, межсетевые экраны могут быть настроены на пропускание инкапсулированного в UDP трафика IPsec, но не трафика ESP/AH без инкапсуляции (или наоборот).

Общей практикой в NAT является трансляция номеров портов TCP и UDP вместе с сетевыми адресами и использование номера порта во входящих пакетах для определения внутреннего узла, которому пакет адресован. По этой причине, несмотря на то, что пакеты IKE **должны** передаваться в порт (из порта) UDP с номером 500 или 4500, они **должны** восприниматься из любого порта, а отклики **должны** передаваться в тот порт, откуда поступил пакет. Это обусловлено тем, номера портов могут изменяться при прохождении пакетов через NAT. По этой же причине IP-адреса конечных точек IKE обычно не включаются в элементы данных IKE, поскольку те защищены криптографически и не могут прозрачно изменяться в устройствах NAT.

Порт 4500 зарезервирован для инкапсулированных в UDP пакетов ESP и IKE. Конечная точка IPsec, обнаружившая NAT между собой и корреспондентом (как описано ниже), **должна** передавать весь последующий трафик из порта 4500, для которого устройствам NAT не следует требуется специальной обработки (как и для порта 500).

Инициатор может использовать порт 4500 для IKE и ESP, независимо от информации о наличии NAT в момент организации соединения IKE. Когда любая из сторон использует порт 4500, инкапсуляция ESP в UDP не требуется, но необходимо понимание инкапсулированных в UDP пакетов ESP на стороне приема. Инкапсуляцию UDP **недопустимо** использовать для порта 500. Если поддерживается NAT-T[1] (т. е., произошел обмен элементами NAT_DETECTION_*_IP в процессе обмена IKE_SA_INIT), все устройства **должны** быть способны принимать и обрабатывать в любой момент пакеты ESP как с инкапсуляцией в UDP, так и без нее. Каждая сторона может принять решение об использовании инкапсуляции трафика ESP в UDP, независимо от выбора другой стороны. Однако при обнаружении NAT оба устройства **должны** использовать для пакетов ESP инкапсуляцию в UDP.

Ниже приведены конкретные требования для поддержки работы через NAT [NATREQ] (такая поддержка не является обязательной). В этом параграфе требования с уровнем **должны** относятся лишь к устройствам с поддержкой работы через NAT.

- Инициатор и ответчик IKE **должны** включить в свои пакеты IKE_SA_INIT элементы Notify типа NAT_DETECTION_SOURCE_IP и NAT_DETECTION_DESTINATION_IP. Эти элементы могут использоваться для детектирования присутствия NAT между хостами и определения стороны, расположенной за NAT. Элементы размещаются в пакетах IKE_SA_INIT после Ni и Nr (перед необязательным элементом CERTREQ).

- Данными, связанными с уведомлением NAT_DETECTION_SOURCE_IP, является сигнатура SHA-1 для значений SPI (в порядке их следования в заголовке), адреса IP и номера порта, откуда был передан пакет.

  Элементов NAT_DETECTION_SOURCE_IP **может** быть более одного, если отправитель не знает, которая из подключенных сетей будет использоваться для передачи пакетов.

- Данными, связанными с уведомлением NAT_DETECTION_DESTINATION_IP, является сигнатура SHA-1 для значений SPI (в порядке их следования в заголовке), адреса IP и номера порта, куда был передан пакет.

- Получатель уведомления NAT_DETECTION_SOURCE_IP или NAT_DETECTION_DESTINATION_IP **может** сравнить полученные значения с сигнатурой SHA-1 для SPI, адресом IP и номером порта отправителя или получателя (соответственно). При несоответствии получателю пакета **следует** включить работу через NAT. В случаях, когда хэш NAT_DETECTION_SOURCE_IP не совпадает для всех полученных данных NAT_DETECTION_SOURCE_IP, получатель **может** отвергнуть попытку организации соединения, если работа через NAT не поддерживается. Несоответствие хэша NAT_DETECTION_DESTINATION_IP означает, что система, получившая данные NAT_DETECTION_DESTINATION_IP, расположена за NAT и этой системе **следует** начать передачу пакетов keepalive, как определено в [UDPENCAPS], система **может** также отвергнуть попытку организации соединения, если работа через NAT не поддерживается.

- Если ни один из полученных элементов NAT_DETECTION_SOURCE_IP не соответствует ожидаемому IP-адресу и порту отправителя из заголовка IP содержащего данные пакета, это означает, что отправившая данные система расположена за транслятором NAT (т. е., кто-то на пути меняет адрес отправителя в исходном пакете на адрес устройства NAT). В таких случаях получившей данные системе следует разрешить динамическое обновление IP-адресов других систем, как описано ниже.

- Инициатор IKE **должен** проверить элементы NAT_DETECTION_SOURCE_IP и NAT_DETECTION_DESTINATION_IP, если они имеются 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 4 октета с нулевыми значениями в начале заголовка IKE размещаются сразу после заголовка UDP. При туннелировании пакетов ESP через порт UDP 4500 заголовок ESP следует сразу после заголовка UDP. Поскольку первые 4 октета заголовка ESP содержат SPI и корректное значение SPI не может быть равно 0, это позволяет во всех случаях различать сообщения ESP и IKE.

- Реализации **должны** обрабатывать инкапсулированные в UDP принятые пакеты ESP даже в тех случаях, когда наличие NAT не обнаружено.

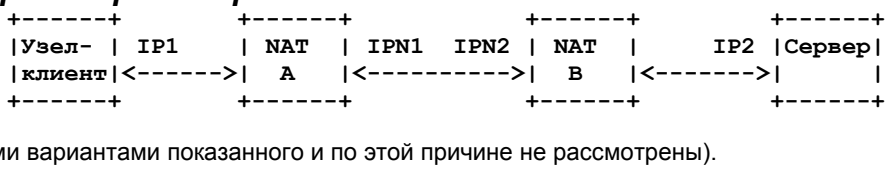
- Исходные IP-адреса отправителя и получателя, которые нужны в транспортном режиме для расчета контрольной суммы пакетов TCP и UDP (см. [UDPENCAPS]), извлекаются из селекторов трафика, связанных с этим обменом. При работе через NAT селекторы трафика **должны** содержать в точности один адрес IP, который используется в качестве исходного адреса. Этот вопрос более подробно рассмотрен в параграфе 2.23.1.

[1] Network Address Translation Traversal - прохождение через NAT.

- В некоторых случаях устройство NAT может удалить отображения, которые продолжают использоваться (например, интервал keepalive слишком велик или транслятор NAT перезагружен). Хост будет видеть это, как будто он получил пакет, прошедший проверку защиты целостности, но имеющий адрес и/или порт, отличающийся от связанного с SA в проверенном пакете. Когда обнаруживается такой прошедший проверку пакет, а хост не поддерживает других методов восстановления типа IKEv2 MOBIKE[1] [MOBIKE] и не находится за NAT, **следует** передавать все пакеты (включая повторы) в IP-адрес и порт из проверенного пакета, а также **следует** сохранить этот адрес и порт, как новую комбинацию для SA (т. е., **следует** динамически обновить адрес). Находящемуся за NAT хосту **не следует** выполнять такое динамическое обновление адреса, если проверенный пакет имеет другой адрес/порт, поскольку это может открывать возможность организации DoS-атаки (позволяя атакующему разорвать соединение, отправив единственный пакет). Динамическое обновление адреса следует проводить только в ответе на новый пакет, поскольку в противном случае атакующий сможет вернуться к адресам из старых, повторно использованных пакетов. По этой причине динамическое обновление адресов может выполняться безопасно лишь при включенной защите от повторного использования пакетов. При использовании IKEv2 с MOBIKE описанное выше динамическое обновление будет конфликтовать с используемым MOBIKE способом восстановления для таких ситуаций (см. параграф 3.8 в [MOBIKE]).

2.23.1. Работа через NAT в транспортном режиме

Транспортный режим, используемый с NAT Traversal, требует специальной обработки селекторов трафика, применяемых в IKEv2. Полный сценарий показан на рисунке (возможны другие сценарии, которые являются упрощенными вариантами показанного и по этой причине не рассмотрены).



В этом сценарии используется два транслятора адресов - NAT A и NAT B. NAT A является динамическим транслятором, который отображает клиентские адреса отправителя IP1 в IPN1. NAT B - статический транслятор, настроенный так, что входящие соединения для адреса IPN2 отображаются на адрес шлюза IP2 (т. е. адрес получателя IPN2 отображается на IP2). Это позволяет клиентам подключаться к серверу, обращаясь по адресу IPN2. Транслятору NAT B не обязательно быть статическим, но клиент должен знать, по какому адресу он может обращаться к серверу и он может добиться этого, лишь обращаясь по внешнему адресу NAT B (т. е., IPN2). Если NAT B является статическим транслятором, его адрес можно будет просто указать в конфигурации клиента. Другим вариантом может служить поиск адреса с использованием того или иного протокола (типа DNS), но это выходит за рамки IKEv2.

В этом сценарии клиент и сервер настраиваются на работу в транспортном режим для трафика, исходящего от клиента и адресованного серверу.

Когда клиент начинает создание IKEv2 SA и Child SA для передачи трафика серверу, он может иметь инициирующий (триггерный) пакет с адресом отправителя IP1 и адресом получателя IPN2. В его базе полномочий партнеров (PAD[2]) и SPD должно быть конфигурационное соответствие между этими адресами (или соответствующими шаблонами).

В транспортном режиме используются в точности те же адреса, что в селекторах трафика и внешних заголовках IP пакетов IKE. Для транспортного режима **должен** применяться в точности один адрес IP в элементах данных TSi и TSr. Может применяться множество селекторов трафика, если имеется, например, множество диапазонов портов, которые хочется согласовать, но все элементы TSi должны использовать диапазон IP1-IP1 в качестве адресов IP и все элементы TSr должны иметь адреса из диапазона IPN2-IPN2. Первому селектору трафика TSi и TSr **следует** иметь конкретные значения Traffic Selector, включая протокол и номера портов (как значения из пакета, инициировавшего запрос).

NAT A будет менять адрес отправителя в пакете IKE с IP1 на IPN1, а NAT B будет менять адрес получателя пакета IKE с IPN2 на IP2, поэтому по прибытии пакета на сервер он будет иметь точно такие селекторы трафика, какие были переданы клиентом, но адреса IP в пакете IKE будут заменены на IPN1 и IP2.

Когда сервер получает этот пакет, он обычно просматривает базу данных PAD, описанную в RFC 4301 [IPSECARCH], по значению ID и ищет SPD по значениям Traffic Selector. Поскольку IP1 реально ничего не говорит серверу (это адрес клиента за NAT), бесполезно заниматься поиском по этому адресу в транспортном режиме. С другой стороны, сервер не может знать, разрешает ли его политика транспортный режим, пока не будет найдена соответствующая запись SPD.

В этом случае серверу следует сначала проверить, запрашивал ли транспортный режим инициатор, а после этого провести замену адреса в селекторах трафика. Сначала нужно сохранить старый адрес IP из селектора трафика, который позднее будет применяться для корректировки контрольной суммы (IP-адрес в TSi может сохраняться, как исходный адрес отправителя, а IP-адрес в TSr - как исходный адрес получателя). После этого, если другая сторона была определена, как расположенная за устройством NAT, сервер меняет адрес IP в элементе данных Tsi на IP-адрес из поля отправителя принятого пакета IKE (т. е., меняет в TSi адрес IP1 на IPN1). Если определено нахождение серверной стороны за устройством NAT, сервер меняет адрес IP в элементе данных TSr на IP-адрес из поля получателя в принятом пакете IKE (т. е., IPN2 в TSr меняется на IP2).

После замены адреса в селекторах трафика и заголовке IKE UDP будут совпадать и сервер может выполнить поиск в SPD по новым значениям Traffic Selector. Если запись найдена и разрешает транспортный режим, эта запись используется. Если же найденная запись не позволяет использовать транспортный режим, сервер **может** вернуться к прежним адресам и повторить поиск в базе SPD по исходным селекторам трафика. Если такой поиск даст положительный результат, сервер будет создавать SA в туннельном режиме, используя реальные значения Traffic Selector, переданные другой стороной.

Такая подстановка адресов в транспортном режиме нужна потому, что поиск SPD выполняется с использованием адресов, которые видны локальному хосту. Это также позволяет добавлять записи в базу данных SAD[3] для проверки точек выхода из туннеля и возврата пакетов с использованием адресов, которые видны стеку локальной операционной системы.

---

[1] Mobility and Multihoming — мобильность и многодомность.
[2] Peer Authorization Database - база данных об уполномоченных партнерах.
[3] Security Association Database - база данных о защищенных связях.

В наиболее общем случае SPD сервера будет включать шаблонные записи, соответствующие адресам, но возможно и включение разных записей SPD (например, для разных известных внешних адресов NAT).

После просмотра SPD сервер будет сужать Traffic Selector на основе найденной в SPD записи. Он будет снова использовать селекторы трафика, в которых уже выполнена подстановка адресов, и возвращать селекторы, имеющие IPN1 и IP2 в качестве адресов IP. Сервер может дополнительно сузить диапазоны номеров протоколов и портов, используемые в селекторах трафика. Запись в базе SAD, создаваемая для Child SA, будет иметь адреса, которые видны серверу (IPN1 и IP2).

При получении клиентом отклика от сервера для Child SA он будет выполнять аналогичную обработку. Если создана SA транспортного режима, клиент может сохранить исходные возвращенные селекторы трафика в качестве исходных адресов отправителя и получателя. Он будет заменять адреса IP в селекторах трафика адресами из заголовка IP в пакете IKE, меняя IPN1 на IP1 и IP2 на IPN2. После этого клиент будет использовать селекторы трафика при сравнении SA с переданными селекторами трафика и организации записи SAD.

Правила для прохождения через трансляторы NAT перечислены ниже.

Для клиентов, предлагающих транспортный режим:

- записи TSi **должны** иметь в точности 1 адрес IP, который **должен** соответствовать адресу отправителя IKE SA;
- записи TSr **должны** иметь в точности 1 адрес IP, который **должен** соответствовать адресу получателя IKE SA;
- первым селекторам трафика TSi и TSr **следует** иметь наиболее специфичные Traffic Selector, включая номера протоколов и портов (такие, зак из вызвавшего запрос пакета);
- **может** присутствовать множество элементов TSi и Tsr;
- если для SA выбран транспортный режим (т. е., сервер включил в свой отклик уведомление USE_TRANSPORT_MODE):
  - исходные селекторы трафика сохраняются, как принятые адреса отправителя и получателя;
  - если сервер расположен за транслятором NAT, адрес IP в TSr заменяется удаленным адресом IKE SA;
  - если клиент расположен за транслятором NAT, адрес IP в TSi заменяется локальным адресом IKE SA;
  - выполняется замена адресов до использования этих селекторов трафика до использования в каких-либо целях, за исключением сохранения их исходного содержимого; сюда включается проверка корректности сужения селекторов трафика другой стороной, создание записи SAD и т. п.

Для отвечающих в случае предложения клиентом транспортного режима:

- сохраняются адреса IP исходного селектора трафика, как полученные адреса отправителя и получателя на случай необходимости их восстановления с целью использования, как «реальных адресов отправителя и получателя», описанного в [UDPENCAPS], и расчета контрольных сумм TCP/UDP;
- если клиент расположен за транслятором NAT, адрес IP в TSi заменяется удаленным адресом IKE SA;
- если сервер расположен за транслятором NAT, адрес IP в TSr заменяется локальным адресом IKE SA;
- выполняется поиск в базах PAD и SPD с использованием ID и селекторов трафика с замененными адресами;
- если в SPD не найдено записи или найденная запись не разрешает транспортный режим, восстанавливаются селекторы трафика с обратной подстановкой адресов; выполняется новый поиск в базах PAD и SPD с использованием ID и селекторов трафика с исходными адресами, а также поиск записи SPD для туннельного режима (с целью перехода в этот режим);
- если запись SPD найдена, выполняется обычное сужение селекторов трафика на основе Traffic Selector с подстановкой адресов и записи SPD; полученные в результате селекторы трафика применяются для создания записей SAD и передачи селекторов трафика обратно клиенту.

2.24. Явное уведомление о перегрузке (ECN)

При развертывании туннелей IPsec в соответствии с исходной спецификацией [IPSECARCH-OLD], использование ECN[1] 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. [ECN]). 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 [ECN], а также **должны** реализовать инкапсуляцию и декапсуляцию в соответствии с [IPSECARCH] для предотвращения отбрасывания индикации перегрузки ECN.

2.25. Конфликты при обменах

Поскольку обмены IKEv2 может инициировать любой из партнеров, возможно частичное перекрытие двух обменов, относящихся к одной SA. Это может приводить к ситуациям, когда информация о состоянии SA временно не синхронизирована и партнеры может получить запрос, который он не сможет обработать нормально.

Обычно использование размера окна больше 1 ведет к возникновению более сложных ситуаций, особенно при обработке запросов с нарушением порядка. В этом разделе рассматриваются проблемы, которые могут возникнуть даже при размере окна 1, и рекомендуемые решения.

[1] Explicit Congestion Notification — явное уведомление о насыщении.

Следует передавать уведомление TEMPORARY_FAILURE в тех случаях, когда принятый запрос временно не может быть выполнен (например, по причине смены ключей). При получении партнером уведомления TEMPORARY_FAILURE ему **недопустимо** сразу же повторять запрос, он **должен** дать отправителю время на завершение операции вызвавшей временный отказ. Получатель уведомления **может** повторить запрос один или множество раз в течение нескольких минут. Если продолжают поступать уведомления TEMPORARY_FAILURE для той же IKE SA по истечении нескольких минут, **следует** принять решение о наличии рассинхронизации данных о состоянии и закрыть IKE SA.

Следует передавать уведомление CHILD_SA_NOT_FOUND при получении запроса на смену ключей для несуществующей связи Child SA. Связь SA, для которой инициатор пытается сменить ключ, указывается полем SPI в элементе Notify, которое копируется из поля SPI в уведомлении REKEY_SA. При получении уведомления CHILD_SA_NOT_FOUND **следует** удалить Child SA (если она еще существует) и отправить запрос на создание новой Child SA с нуля (если Child SA еще не существует).

2.25.1. Конфликты во время смены ключей или закрытия дочерних SA

Если пратнер получает запрос на смену ключей Child SA, для которой в данный момент выполняется попытка завершения, ему **слудет** ответить сообщением TEMPORARY_FAILURE. Если такой запро поступает во время происходящей уже замены ключей, на него **следует** ответить обычным способом, а также **следует** подготовиться к последующему закрытию избыточных SA на базе значений nonce (см. параграф 2.8.1). Если приходит запрос смены ключей для отсутствующей Child SA, на него **следует** ответить сообщением CHILD_SA_NOT_FOUND.

При получении запроса на закрытие Child SA, для которой уже начата попытка закрытия, на него **следует** ответить без элемента Delete (см. параграф 1.4.1). Если такой запрос приходит для Child SA, в которой меняются ключи, на него **следует** отвечать обычным путем с элементом Delete. При получение запроса на закрытие отсутствующей Child SA **следует** отвечать без элемента Delete.

При получении запроса на смену ключей IKE SA, в которой происходит создание, смена ключей или закрытие Child SA в данной IKE SA **следует** отвечать на него сообщением TEMPORARY_FAILURE.

2.25.2. Конфликты во время замены ключей или закрытия IKE SA

При получении запроса на смену ключей IKE SA, в которой уже идет процесс смены ключей, не него **следует** отвечать обычным способом, а также **следует** приготовиться к последующему закрытию избыточных SA и переносу наследуемых Child SA на основе значений nonce (см. параграф 2.8.2). При получении такого запроса для закрываемой в данный момент IKE SA на него **следует** отвечать сообщением TEMPORARY_FAILURE.

При получении запроса на закрытие IKE SA, в которой происходит смена ключей на него **следует** отвечать обычным способом, забывая о запросе на смену ключей. При получении запроса на закрытие IKE SA, которая уже закрывается на него **следует** отвечать обычным способом, забывая имеющийся запрос на закрытие.

При получении запроса на создание или смену ключей Child SA во время смены ключей IKE SA на него **следует** отвечать сообщением TEMPORARY_FAILURE. При получении запроса на удаление Child SA во время смены ключей IKE SA **следует** отвечать обычным способом с элементом Delete.

3. Форматы заголовков и данных

В таблицах этого раздела некоторые криптографические примитивы и конфигурационные атрибуты обозначены, как не заданные (UNSPECIFIED). Это элементы, для которых нет известных спецификаций и, следовательно, взаимодействие с ними в настоящее время невозможно. В будущих спецификациях может быть описано их использование, но пока таких спецификаций не выпущено, реализациям **не следует** пытаться использовать элементы, отмеченные, как UNSPECIFIED, если нужно обеспечить интероперабельность реализации.

3.1. Заголовок IKE

Сообщения IKE используют протокол UDP через порты 500 и/или 4500, передается 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 идентифицируются в порядке их следования в сообщении IKE путем 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 защищенной связи IKE. Следовательно, один экземпляр IKE может мультиплексировать различные сессии с множеством партнеров, а также множество сессий с одним партнером.

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

Рисунок 4 показывает формат заголовка IKE.

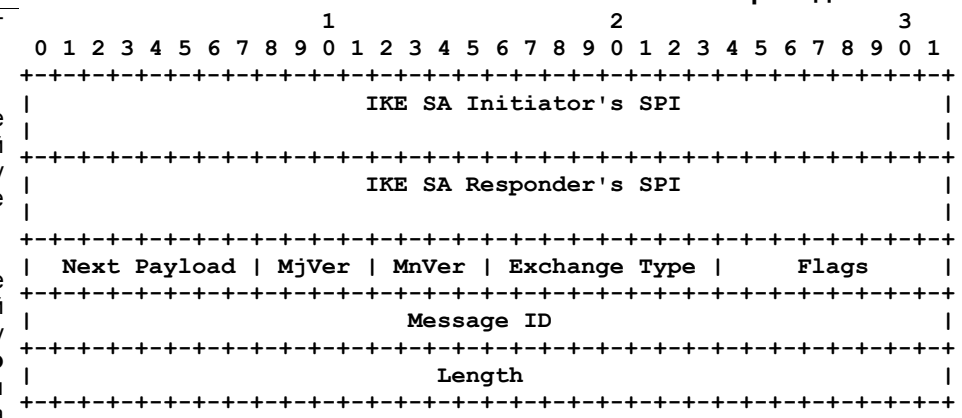


Рисунок 4: Формат заголовка IKE

Initiator's SPI (8 октетов)
Значение, выбранное инициатором для уникальной идентификации IKE Security Association. Нулевое значение **недопустимо**.

Responder's SPI (8 октетов)
Значение, выбранное ответчиком для уникальной идентификации IKE Security Association. Это поле **должно** иметь значение 0 в первом сообщении начального обмена IKE (включая повторы этого сообщения, содержащие cookie).

Next Payload (1 октет)
Указывает тип элемента данных, расположенного сразу после заголовка. Форматы и значения всех типов описаны ниже.

Major Version (4 бита)
Указывает 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, возвращая уведомление INVALID_MAJOR_VERSION, как описано в параграфе 2.5.

Minor Version (4 бита)
Указывает 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)
Указывает тип обмена, который будет применяться. Это значение ограничивает тип элементов данных, передаваемых в каждом сообщении обмена. Приведенные в таблице справа значения типов были определены на момент публикации RFC 4306. Другие значения могли быть добавлены с тех пор и могут добавляться после публикации этого документа. Актуальный набор значений можно найти в [IKEV2IANA].

Тип обмена	Значение
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37

Flags (1 октет)
Определяет конкретные опции, установленные для данного сообщения. Присутствие опции указывается установкой соответствующего бита в поле флагов, показанном ниже.

```
+-+-+-+-+-+-+-+-+
|X|X|R|V|I|X|X|X|
+-+-+-+-+-+-+-+-+
```

В приведенных ниже описаниях установленные флаги имеют значение 1, сброшенные — 0. Флаги, обозначенные символом X **должны** быть сброшены при передаче, а на приемной стороне **должны** игнорироваться.

R (Response)
Этот флаг указывает, что сообщение является откликом на сообщение с таким же Message ID. Флаг **должен** быть сброшен во всех запросах и **должен** устанавливаться во всех откликах. Конечным точкам IKE **недопустимо** генерировать отклики на сообщения, помеченные как отклики (с одним исключением, описанным в параграфе 2.21.2).

V (Version)
Этот флаг показывает, что передающая сторона может поддерживать более высокое значение старшего номера версии протокола, нежели указано в поле Major version. Реализации IKEv2 **должны** сбрасывать этот бит при передаче и игнорировать его в принимаемых сообщениях.

I (Initiator)
Этот бит **должен** устанавливаться в сообщениях, передаваемых исходным инициатором IKE SA, и **должен** сбрасываться в сообщениях от исходного ответчика. Флаг используется получателем для того, чтобы определить, какие 8 октетов SPI были созданы получателем. Значение бита меняется в соответствии с тем, кто инициировал последнюю смену ключей IKE SA.

Message ID (4 октета, целое число без знака)
Идентификатор сообщения, используемый для управления повтором передачи потерянных пакетов и определения соответствия между запросами и откликами. Идентификатор играет важную роль в безопасности протокола, поскольку служит для предотвращения атак с повторным использованием пакетов (см. параграфы 2.1 и 2.2).

Length (4 октета, целое число без знака)
Размер всего сообщения (заголовок и элементы данных) в октетах.

3.2. Базовый заголовок элемента данных

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

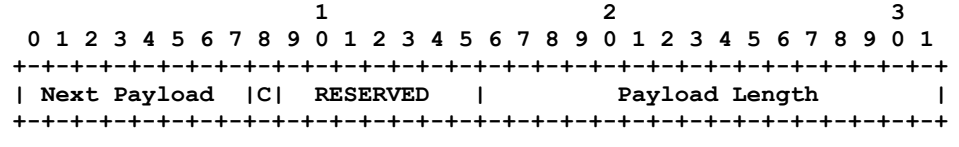


Рисунок 5. Базовый заголовок элемента данных

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 формате дополнительных элементов. В заголовке элемента Encrypted поле Next Payload устанавливается в соответствии с типом первого вложенного элемента (вместо 0), а в поле Next Payload последнего вложенного элемента устанавливается 0. Значения типов элементов данных показаны в таблице. Эти значения были определены на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Значения 1 - 32 не были распределены для того, чтобы не возникало перекрытия с кодами, определенными для IKEv1.

Тип следующего элемента данных	Обозначение	Значение
Нет		0
Защищенная связь	SA	33
Обмен ключами	KE	34
Идентификация - инициатор	IDi	35
Идентификация - ответчик	IDr	36
Сертификат	CERT	37
Запрос сертификата	CERTREQ	38
Аутентификация	AUTH	39
Nonce	Ni, Nr	40
Уведомление	N	41
Удаление	D	42
Идентификатор производителя	V	43
Селектор трафика - инициатор	TSi	44
Селектор трафика - ответчик	TSr	45
Зашифрованные и аутентифицированные	SK	46
Конфигурация	CP	47
Расширяемая аутентификация	EAP	48

Critical (1 бит)

Это поле **должно** иметь значение 0, если отправитель хочет, чтобы получатель пропустил этот элемент в случае непонимания им кода в поле Next Payload предыдущего элемента. Поле **должно** иметь значение 1, если отправитель хочет, чтобы получатель целиком отвергал сообщение с непонятным типом элемента данных. Если получатель понимает тип элемента, он **должен** игнорировать этот флаг. Это поле **должно** устанавливаться в 0 для определенных 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 всех определенных здесь элементов заключается в том, что все реализации **должны** поддерживать все типы элементов, определенные в этой спецификации, и, следовательно, должны игнорировать значение флага Critical. Предполагается, что пропускаемые элементы будут иметь корректные значения полей Next Payload и Payload Length. Дополнительная информация об этом флаге приведена в параграфе 2.5.

RESERVED (7 битов)

Должен устанавливаться в 0 при передаче и игнорироваться на приемной стороне.

Payload Length (2 октета, целое число без знака)

Размер текущего элемента данных в октетатх с учетом базового заголовка элемента данных.

Многие элементы данных содержат резервные (RESERVED) поля. Некоторые элементы данных в IKEv2 (и в IKEv1) не выравниваются по 4-октетным границам.

3.3. Элемент данных SA

Элементы данных защищенной связи, обозначаемые в этом документе SA, служат для согласования атрибутов защищенной 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 включает один протокол 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 свое 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 ESP с комбинацией (3DES и HMAC_MD5) или с комбинацией (AES и HMAC_SHA1).

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

Структура Proposal включает Proposal Num (номер предложения) и идентификатор протокола IPsec. Каждая структура **должна** использовать номер предложения, увеличенный на 1 по сравнению со значением в предыдущей структуре. Первый элемент Proposal в SA инициатора **должен** иметь Proposal Num = 1. Одной из причин использования множества предложений является обеспечение возможности предложить использования стандартных и комбинированных шифров. Комбинированные шифры обеспечивают шифрование и защиту целостности на основе одного алгоритма и **должны** не предлагать алгоритма защиты целостности или предлагать алгоритм NONE (первый вариант является **рекомендуемым**). Если инициатор хочет предложить комбинированные и стандартные шифры, он должен включить два предложения — одно включает все комбинированные алгоритмы, а другое — все обычные алгоритмы в комбинации с алгоритмами защиты целостности. Например, одно такое предложение будет включать две

структуры Proposal. Первая структура Proposal 1 предлагает ESP с ключами AES-128, AES-192, AES-256 в режиме CBC[1] и алгоритмом защиты целостности HMAC-SHA1-96 или XCBC-96, а вторая структура Proposal 2 предлагает AES-128 или AES-256 в режиме GCM с 8-октетным значением контроля четности (ICV[2]). Оба предложения позволяют, но не требуют использовать расширенные порядковые номера ESN[3]. Вид предложений представлен на рисунке.

```
Элемент SA
   |
   +--- Proposal #1 ( Proto ID = ESP(3), SPI size = 4,
   |     |            7 преобразований,      SPI = 0x052357bb )
   |     |
   |     +-- Преобразование ENCR ( Name = ENCR_AES_CBC )
   |     |     +-- Атрибут ( Key Length = 128 )
   |     |
   |     +-- Преобразование ENCR ( Name = ENCR_AES_CBC )
   |     |     +-- Атрибут ( Key Length = 192 )
   |     |
   |     +-- Преобразование ENCR ( Name = ENCR_AES_CBC )
   |     |     +-- Атрибут ( Key Length = 256 )
   |     |
   |     +-- Преобразование INTEG ( Name = AUTH_HMAC_SHA1_96 )
   |     +-- Преобразование INTEG ( Name = AUTH_AES_XCBC_96 )
   |     +-- Преобразование ESN ( Name = ESN )
   |     +-- Преобразование ESN ( Name = не ESN )
   |
   +--- Proposal #2 ( Proto ID = ESP(3), размер SPI = 4,
         |            4 преобразования,      SPI = 0x35a1d6f2 )
         |
         +-- Преобразование ENCR ( Name = AES-GCM с 8 октетами ICV )
         |     +-- Атрибут ( Key Length = 128 )
         |
         +-- Преобразование ENCR ( Name = AES-GCM с 8 октетами ICV )
         |     +-- Атрибут ( Key Length = 256 )
         |
         +-- Преобразование ESN ( Name = ESN )
         +-- Преобразование ESN ( Name = не ESN )
```

За каждой структурой Proposal/Protocol следует одна или множество структур преобразований. Число разных преобразований обычно определяется элементом Protocol. Протокол AH обычно имеет два преобразования - расширенные порядковые номера (ESN[4]) и алгоритм контроля целостности. ESP обычно имеет три преобразования - ESN, алгоритм шифрования и алгоритм контроля целостности. IKE обычно имеет четыре преобразования - группа Diffie-Hellman, алгоритм контроля целостности, алгоритм PRF и алгоритм шифрования. Для каждого элемента Protocol набор разрешенных преобразований указывается значениями Transform ID после заголовка каждого преобразования.

Если имеется множество преобразований с одним значением Transform Type, они должны комбинироваться в одно предложение с помощью операции **ИЛИ**[5]. 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 (для HMAC_MD5 и HMAC_SHA). Это обеспечивает эффективное предложение четырех 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 - 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).

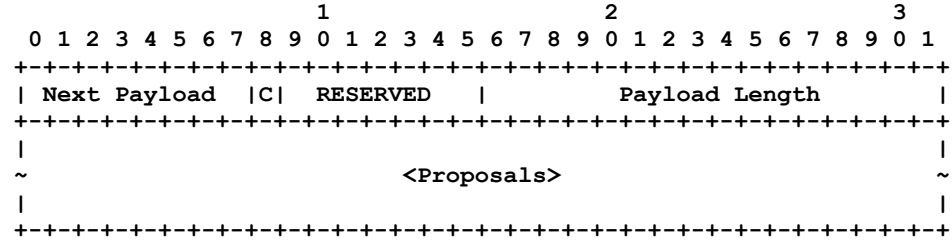


Рисунок 6. Элемент данных SA

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

Proposals (переменный размер)

Одна или множество субструктур Proposal.

Идентификатор типа для элемента данных SA имеет значение 33.

[1]Cipher Block Chaining — цепочка шифрованных блоков.
[2]Integrity Check Value.
[3]Extended Sequence Number.
[4]Extended Sequence Number.
[5]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.*

3.3.1. Субструктура Proposal

Last Substruc (1 октет)

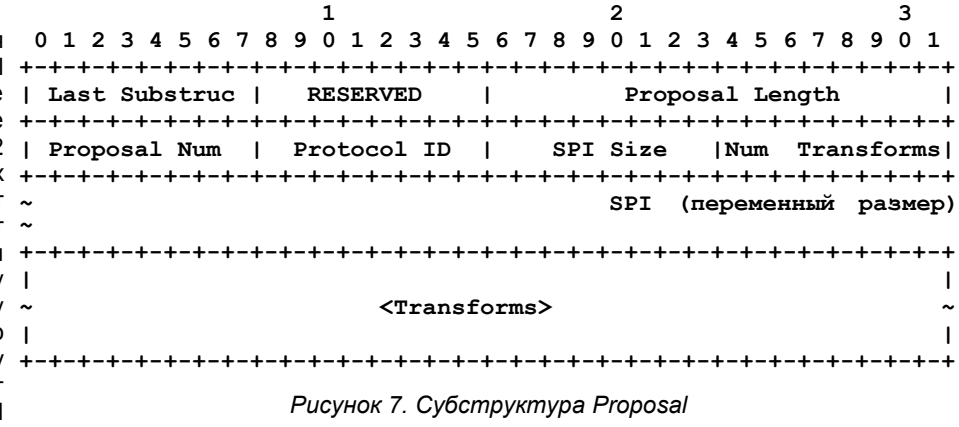


Рисунок 7. Субструктура Proposal

Указывает, является ли данная субструктура Proposal последней в SA. Это поле имеет значение 0 в случае последней субструктуры и 2 при наличии дополнительных субструктур Proposal. Этот синтаксис унаследован от ISAKMP, но не является требуемым, поскольку последнюю субструктуру Proposal можно идентифицировать по размеру SA. Значение (2) соответствует элементу данных типа Proposal в IKEv1, а первые 4 октета субструктуры Proposal организованы так, чтобы выглядеть подобно заголовку элемента данных.

RESERVED (1 октет)

Должно устанавливаться в 0 при передаче и игнорироваться при получении.

Proposal Length (2 октета, целое число без знака)

Размер данной субструктуры Proposal с учетом всех следующих за ним преобразований и атрибутов.

Proposal Num (1 октет)

Когда предложение вносится, первое из предложений в данных SA **должно** иметь номер 1, а последующие **должны** иметь номера на 1 больше своего предшественника (указывается оператором OR между предложениями). Когда предложение принимается, его номер в элементе данных SA **должен** совпадать с номером, с которым оно было передано.

Protocol ID (1 октет)

Задает идентификатор протокола IPsec для текущего согласования. Определенные на момент публикации RFC 4306 значения идентификаторов показаны в таблице. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Протокол	Идентификатор
IKE	1
AH	2
ESP	3

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).

Num Transforms (1 октет)

Указывает число преобразований в этом предложении.

SPI (переменный размер)

Значение SPI передающей стороны. Даже при SPI Size не кратном 4 октетам заполнение для этого поля не применяется. При SPI Size = 0 это поле отсутствует в элементе данных SA.

Transforms (переменный размер)

Одна или множество субструктур Transform.

3.3.2. Субструктура Transform

Last Substruc (1 октет)

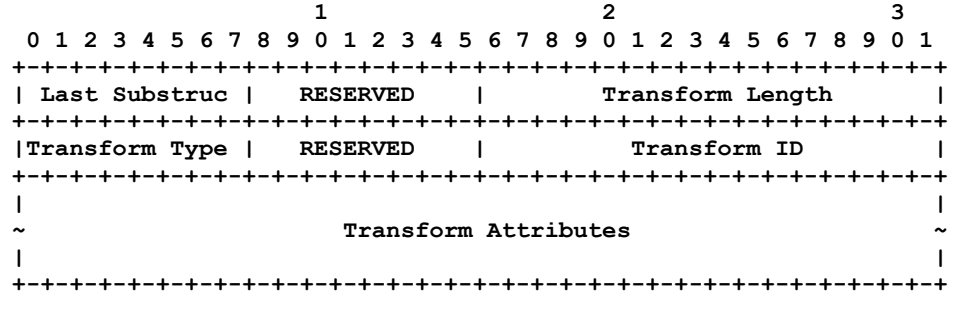


Рисунок 8. Субструктура Transform

Указывает, является ли данная субструктура Transform последней в Proposal. Это поле имеет значение 0 в случае последней субструктуры и 3 при наличии дополнительных субструктур Transform. Синтаксис поля унаследован от ISAKMP, но не является требуемым, поскольку последнюю субструктуру Transform можно идентифицировать по размеру Proposal. Значение (3) соответствует элементу данных типа Transform в IKEv1, а первые 4 октета субструктуры Transform организованы так, чтобы выглядеть подобно заголовку элемента данных.

RESERVED

Должно устанавливаться в 0 при передаче и игнорироваться при получении.

Transform Length

Размер данной субструктуры Transform с учетом всех следующих за ним преобразований и атрибутов.

Transform Type (1 октет)

Тип преобразования, задаваемого этой субструктурой. Различные протоколы поддерживают разные значения Transform Type. Для некоторых протоколов отдельные преобразования могут быть необязательными. Если преобразование не является обязательным, а инициатор желает внести предложение без него, такое преобразование просто опускается. Если инициатор хочет передать решение вопроса использования необязательного преобразования ответчику, он включает эту субструктуру с Transform ID = 0 в качестве одной из опций.

Transform ID (2 октета)

Указывает конкретный экземпляр Transform Type, который будет предложен.

Значения Transform Type, приведенные в таблице справа, отражают типы, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа.

Актуальные значения следует смотреть в [IKEV2IANA].

Описание	Тип	Применение
Алгоритм шифрования (ENCR)	1	IKE и ESP
Псевдослучайная функция (PRF)	2	IKE
Алгоритм защиты целостности (INTEG)	3	IKE*, AH, опционально в ESP
Группа Diffie-Hellman (D-H)	4	IKE, может использоваться также в AH и ESP
Расширенные порядковые номера (ESN)	5	AH и ESP

(*) Согласование алгоритма защиты целостности является обязательным для формата Encrypted, описанного в этом документе. Например, [AEAD] задает дополнительные форматы, основанные на аутентифицированном шифровании, где отдельный алгоритм защиты целостности не согласуется.

Для преобразований типа 1 (Transform Type 1, алгоритм шифрования) значения идентификаторов преобразования (Transform ID) приведены в таблице слева, включающей идентификаторы, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Имя	Номер	Документ
ENCR_DES_IV64	1	Нет спецификации
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	Нет спецификации
ENCR_3IDEA	8	Нет спецификации
ENCR_DES_IV32	9	Нет спецификации
ENCR_NULL	11	RFC2410
ENCR_AES_CBC	12	RFC3602
ENCR_AES_CTR	13	RFC3686

Для преобразований типа 2 (Transform Type 2, псевдослучайная функция) значения Transform ID приведены в таблице справа, включающей идентификаторы, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Имя	Номер	Документ
PRF_HMAC_MD5	1	RFC2104, [MD5]
PRF_HMAC_SHA1	2	RFC2104, [FIPS.180-4.2012]
PRF_HMAC_TIGER	3	Нет спецификации

Для преобразований типа 3 (Transform Type 3, алгоритм защиты целостности) значения Transform ID приведены в таблице справа, включающей идентификаторы, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Имя	Номер	Документ
NONE	0	
AUTH_HMAC_MD5_96	1	RFC2403
AUTH_HMAC_SHA1_96	2	RFC2404
AUTH_DES_MAC	3	Нет спецификации
AUTH_KPDK_MD5	4	Нет спецификации
AUTH_AES_XCBC_96	5	RFC3566

Для преобразований типа 4 (Transform Type 4, группа Diffie-Hellman) значения Transform ID приведены в таблице слева, включающей идентификаторы, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Имя	Номер	Документ
NONE	0	
768-битовый MODP	1	Приложение B
1024-битовый MODP	2	Приложение B
1536-битовый MODP	5	[ADDGROUP]
2048-битовый MODP	14	[ADDGROUP]
3072-битовый MODP	15	[ADDGROUP]
4096-битовый MODP	16	[ADDGROUP]
6144-битовый MODP	17	[ADDGROUP]
8192-битовый MODP	18	[ADDGROUP]

Хотя протоколы ESP и AH не включают напрямую обмена Diffie-Hellman, группа DH **может** быть согласована для Child SA. Это позволяет партнерам использовать метод Diffie-Hellman в обмене CREATE_CHILD_SA для обеспечения совершенной защиты генерируемых ключей Child SA.

Отметим, что указанные в таблице группы MODP Diffie-Hellman не требуют выполнения каких-либо специальных тестов на пригодность, однако другие типы групп (например, группы эллиптических кривых или группы MODP с малыми подгруппами) требуют тех или иных специальных проверок для из безопасного применения (см. параграф Additional Diffie-Hellman Tests for IKEv2 в [RFC6989]).

Для преобразований типа 5 (Transform Type 5, расширенные порядковые номера ESN) значения Transform ID приведены в таблице справа, включающей идентификаторы, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Имя	Номер
Нет ESN	0
ESN	1

Отметим, что инициатор, поддерживающий ESN, обычно будет включать два преобразования ESN со значениями 0 и 1 в своих предложениях. Предложения, включающие только приобразование ESN со значением 1, означате отказ от применения обычных (не расширенных) порядковых номеров.

С момента публикации RFC 4306 было добавлено множество значения Transform Type, которые можно найти в реестре IANA IKEv2.

3.3.3. Приемлемые типы преобразований для протоколов

Число и тип преобразований в элементах SA зависит от протокола в самой SA. Элемент данных SA, предлагающий организацию SA, сопровождается обязательными и опциональными типами преобразований. Соответствующая спецификации реализация **должна** понимать все обязательные и опциональные типы для поддерживаемых ей протоколов (хотя она не обязана принимать предложения с неприемлемыми шифрвми). Предложение **может** опускать необязательные типы, если единственным воспринимаемым значением является NONE.

3.3.4. Обязательные Transform ID

Указание шифронаборов, которые **должно** и **следует** поддерживать для интероперабельности, было удалено из этого документа, поскольку стало ясно, что эти списки будут меняться чаще, нежели обновляется документ. На момент публикации этого документа такие шифронаборы были указаны в [RFC4307], но следует отметить, что этот список может быть обновлен в будущем, а другие RFC могут задавать другие наборы шифров.

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

(*) Согласование алгоритма защиты целостности является обязательным для формата Encrypted, описанного в этом документе. Например, [AEAD] задает дополнительные форматы, основанные на аутентифицированном шифровании, где отдельный алгоритм защиты целостности не согласуется.

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.

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

Каждое преобразование в элементе SA может включать атрибуты, меняющие или дополняющие спецификацию преобразования. Набор приемлемых атрибутов зависит от преобразования. В настоящее время определен единственный тип атрибута — размер ключа (Key Length) используемый для некоторых алгоритмов шифрования с переменным размером ключей (см. ниже).

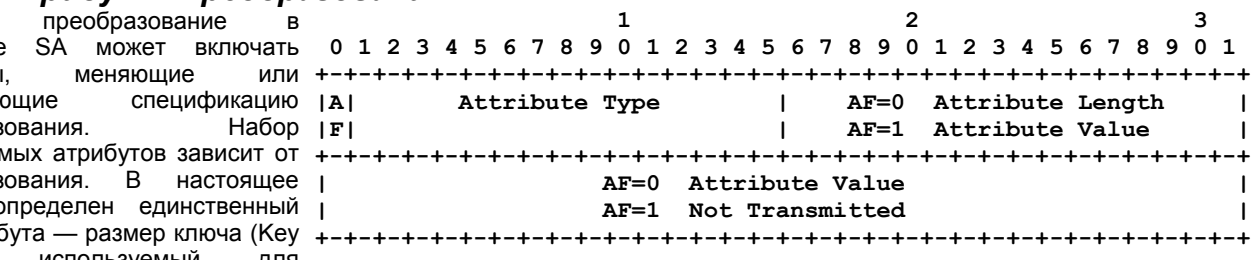


Рисунок 9. Атрибуты преобразования

Эти атрибуты представляют собой пары «тип-значение» и определены ниже. Атрибуты могут иметь значение фиксированного (2 октета) или переменного размера. В последнем случае для представления атрибута используется формат «тип-размер-значение».

Attribute Format (AF) (1 бит)

Указывает использование для данных атрибута формата TLV[1] или более короткого TV[2]. Нулевое значение флага AF говорит о формате TLV, а 1 - о формате TV (2-байтовое значение).

Attribute Type (15 битов)

Уникальный идентификатор типа атрибута (см. ниже).

Attribute Value (переменный размер)

Значение атрибута указанного типа. Если бит AF сброшен (0), это поле имеет переменный размер, указанный полем Attribute Length. При установленном (1) флаге AF поле Attribute Value имеет размер 2 октета.

Единственным определенным в настоящее время атрибутом является размер ключа (Key Length) и для него используется фиксированный размер. Спецификации атрибутов переменной длины включены только для будущих расширений. 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. Отметим, что это отличается от IKEv1 тем, что повышается гибкость и упрощается создание сообщений, но несколько усложняется их разбор.

В таблице указано значение, определенное на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Тип атрибута	Значение	Формат атрибута
Key Length (в битах)	14	TV

Значения 0 - 13 и 15 - 17, использовалишь в похожем контексте IKEv1 и их не следует выделять без наличия совпадений.

Атрибут Key Length указывает размер ключа в битах (**должен** использоваться сетевой порядок байтов) для некоторых преобразований, как описано ниже.

[1] Type/Length/Value — тип-размер-значение.
[2] Type/Value — тип-значение.

- Атрибут Key Length **недопустимо** использовать в преобразованиях с фиксированным размером ключа. Например, к таким преобразованиям относятся ENCR_DES, ENCR_IDEA, а также все преобразования Type 2 (псевдослучайная функция) и Type 3 (защита целостности), описанные в этом документе.

- Для некоторых преобразований атрибут Key Length **должен** включаться всегда (пропускать атрибут не разрешается и преобразования без него **должны** отвергаться). Примерами могут служить преобразования ENCR_AES_CBC и ENCR_AES_CTR.

- В некоторых преобразованиях с переменным размером ключей устанавливается используемый по умолчанию размер. Примерами таких преобразований являются ENCR_RC5 и ENCR_BLOWFISH.

Примечание для разработчиков. Для повышения уровня интероперабельности и поддержки независимого обновления конечных точек реализациям этого протокола **следует** воспринимать значения, которые на их взгляд повышают уровень защиты. Например, если узел настроен на восприятие шифров с переменным размером ключа при длине ключа X битов и предлагается более длинный ключ, реализации **следует** принимать такое предложение, если она может работать с более длинным ключом.
Поддержка такой возможности позволяет ответчику выразить концепцию «не ниже» некоторого уровня защиты — «размер ключа не менее X битов для шифра Y». Однако, поскольку атрибут всегда возвращается неизменным (см. следующий параграф), инициатору, желающему воспринимать разные размеры ключей, следует включать множество преобразования одного типа, каждое из которых имеет свой атрибут Key Length.

3.3.6. Согласование атрибутов

При согласовании защищенной 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 одно из них. Если выбранное предложение включает 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.

Если ответчик получает предложение с непонятным Transform Type или в предложении отсутствует обязательный тип преобразования, такое предложение **должно** рассматриваться, как неприемлемое, однако другие предложения из того же элемента SA обрабатываются обычным путем. Аналогично, если ответчик получает преобразование непонятного типа или с непонятным значением Transform Attribute, он **должен** рассматривать такое преобразование, как неприемлемое, сохраняя обычную обработку других прелобразований с тем же Transform Type. Это позволит определят в будущем новые значения Transform Type и Transform Attribute.

Согласование групп Diffie-Hellman имеет некоторые особенности. SA включает предлагаемые атрибуты и открытое значение Diffie-Hellman (KE[1]) 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 ответчик выберет предложение с другой группой DH (отличной от NONE), он 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. Если одним из предложений для группы DH является NONE и ответчик выбирает предложение, он **должен** игнорировать элемент KE инициатора и не включать KE в свой отклик.

3.4. Элемент Key Exchange

Элемент данных Key Exchange (KE) служит для обмена открытыми значениями Diffie-Hellman в процессе обмена ключами Diffie-Hellman. Элемент Key Exchange состоит из базового заголовка данных IKE, за которым следует открытое значение Diffie-Hellman.

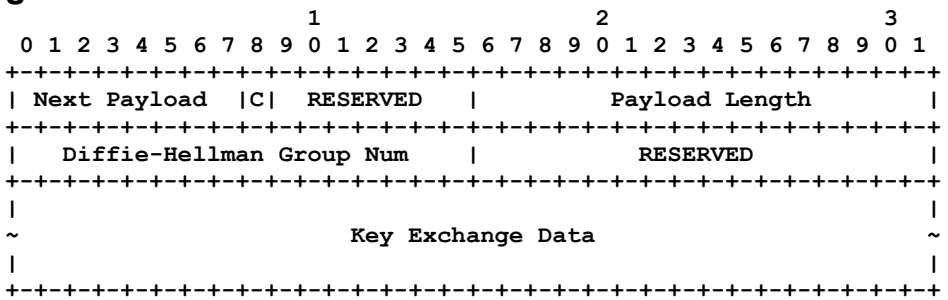


Рисунок 10. Формат элемента Key Exchange

KE создается путем копирования открытого значения Diffie-Hellman в поле Key Exchange Data. Размер открытого значения DH для модульных экспоненциальных групп (MODP[2]) **должен** совпадать с размером простого числа (prime modulus), которое возводится в степень. При необходимости поле заполняется нулями слева (prepend).

Поле Diffie-Hellman Group Num указывает группу DH, в которой будет рассчитываться Key Exchange Data (см. параграф 3.3.2). Поле Diffie-Hellman Group Num **должно** соответствовать группе DH, заданной в предложении элемента данных SA, переданного в том же сообщении, **следует** также обеспечивать соответствие группе DH в первой группе первого предложения, если она указана. Если ни одно из предложений с данном элементе SA не указывает группу DH, наличие элемента KE **недопустимо**. Если выбранное предложение использует другую группу DH (отличную от NONE), сообщение **должно** быть отвергнуто с возвратом элемента Notify типа INVALID_KE_PAYLOAD. См. также параграфы 1.2 и 2.7.

Идентификатор типа для элемента Key Exchange имеет значение 34.

[1]Элемент данных Key Exchange. *Прим. перев.*
[2]Modular exponentiation group.

3.5. Элемент Identification

Элемент данных Identification (ID), обозначаемый в этом документе IDi и IDr, позволяет партнерам 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. При использовании отождествлений типа ID_IPV4_ADDR/ID_IPV6_ADDR в элементах IDi/IDr протокол IKEv2 не требует соответствия этих адресов адресам в заголовках IP пакетов IKEv2 или содержимом элементов TSi/TSr. Информация IDi/IDr используется исключительно для выбора политики и аутентификационных данных, относящихся к другой стороне.

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

База данных PAD[1], как указано в RFC 4301 [IPSECARCH], описывает использование элемента ID в IKEv2 и обеспечивает формальную модель для привязки отождествления в политики в дополнение к службам, связанным более конкретно с исполнением политики. PAD предназначена для обеспечения связи между SPD и управления IKE Security Association (дополнительную информацию можно найти в параграфе 4.4.3 RFC 4301).

Элемент Identification состоит из базового заголовка данных IKE, за которым следуют описанные ниже поля идентификации.

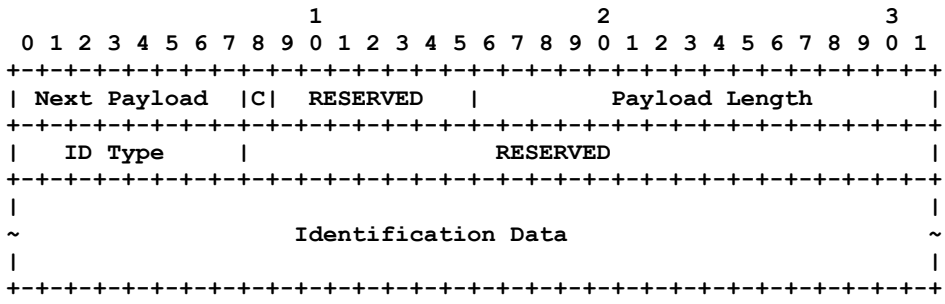


Рисунок 11. Формат данных Identification

ID Type (1 октет)
Задает тип используемой идентификации.

RESERVED
Должно устанавливаться в **0** при передаче и игнорироваться при получении.

Identification Data (переменный размер)
Значение, определяемое полем Identification Type. Размер идентификационных данных рассчитывается из размера в заголовке элемента ID.

Идентификатор типа для элемента Identification имеет значение 35 в случае IDi и 36 в случае IDr.

В таблице приведены значения типов идентификации, выделенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

ID Type	Значение	Описание Identification Data
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.). Все символы ID_FQDN указываются в кодировке ASCII, для доменных имен на других языках применяется синтаксис, определенный в [IDNA] (например, xn--tmonesimerkki-bfbb.example.net).
ID_RFC822_ADDR	3	Строка полного почтового адреса RFC822 (например, jsmith@example.com). В строку **недопустимо** включать символы завершения. С учетом [EAI] реализациям уместно трактовать эту строку, как текст UTF-8, а не ASCII.
ID_IPV6_ADDR	5	Один шестнадцатиоктетный адрес IPv6.
ID_DER_ASN1_DN	9	Двоичное представление в формате DER[2] для ASN.1 X.500 Distinguished Name [PKIX].
ID_DER_ASN1_GN	10	Двоичное представление в формате DER для ASN.1 X.500 GeneralName [PKIX].
ID_KEY_ID	11	Неструктурированный поток октетов, который может использоваться для передачи связанной с производителем информации, требуемой для некоторых фирменных вариантов идентификации.

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 вместо ID_IPV4_ADDR.

EAP [EAP] не требует использования какого-либо конкретного типа идентификаторов, но зачастую EAP применяется с идентификаторами NAI[3], определенными в [NAI]. Хотя идентификаторы NAI похожи на адреса электронной почты (например, joe@example.com), их синтаксис отличается от синтаксиса почтовых адресов [MAILFORMAT]. По этой причине для NAI с компонентов realm (область) **следует** указывать типа ID_RFC822_ADDR. Реализациям ответчиков не нужно предпринимать попыток проверки точного соответствия идентификатора синтаксису [MAILFORMAT], принимая любые, выглядящие разумно NAI. Для идентификаторов NAI без компоненты **следует** указывать тип ID_KEY_ID.

Дополнительная информация о соответствии элементов Identification и содержимого сертификатов PKIX приведена в [RFC4945].

[1]Peer Authorization Database — база данных для аутентификации партнеров.
[2]Distinguished Encoding Rules.
[3]Network Access Identifier — идентификатор доступа в сеть.

3.6. Элемент Certificate

Элемент данных Certificate (CERT) обеспечивает способ передачи сертификатов или других, связанных с идентификацией данных через IKE. Элементы Certificate **следует** включать в обмен, если сертификаты доступны отправителю. Форматы Hash and URL (хэш и ссылка) для элементов Certificate следует применять в тех случаях, когда партнер указал возможность получения информации иным путем 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.

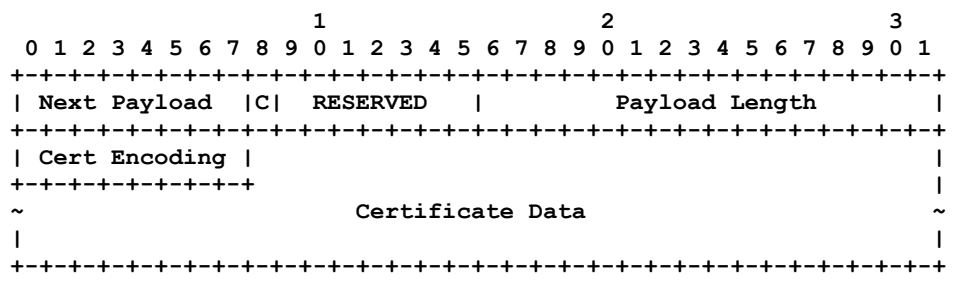


Рисунок 12. Формат элемента Certificate

Поля элемента Certificate описаны ниже.

Сертификат	Код	Описание
PKCS #7 wrapped X.509	1	Не задано
PGP	2	Не задано
DNS Signed Key	3	Не задано
X.509 - подпись	4	
Kerberos Token	6	Не задано
CRL (список отозванных сертификатов)	7	
ARL (список УЦ с прерванными полномочиями)	8	Не задано
SPKI	9	Не задано
X.509 - атрибут	10	Не задано
Raw RSA Key	11	
Хэш и URL сертификата X.509	12	
Хэш и URL связки (bundle) X.509	13	

Certificate Encoding (1 октет)

Это поле указывает тип сертификата или связанной с сертификатом информации в поле Certificate Data. В таблице приведены значения типов сертификатов, выделенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Certificate Data (переменный размер)

Реальный размер сертификационных данных. Тип сертификата задается полем Certificate Encoding field.

Идентификатор типа для элемента Certificate имеет значение 37.

Конкретный синтаксис для некоторых типов сертификатов в этом документе не определен. Ниже приведен список типов сертификатов, для которых синтаксис определен в этой спецификации.

- «X.509 - подпись» содержит DER-представление сертификата X.509, открытый ключ которого служит для проверки элемента AUTH отправителя. Отметим, что при этом варианте представления в случаях, когда нужно передать цепочку сертификатов, используется множество элементов CERT и только первый из них содержит открытый ключ, используемый для проверки элемента AUTH отправителя.
- CRL содержит DER-представление списка отозванных сертификатов X.509.
- Raw RSA Key содержит представление PKCS #1 ключа RSA, т. е., представленную в формате DER структуру RSAPublicKey (см. [RSA] и [PKCS1]).
- Представления «Хэш и URL» позволяют снизить размер сообщений IKE путем замены длинных структур данных 20-октетным хэш-значением SHA-1 (см. [FIPS.180-4.2012]) замененного сертификата и ссылкой (URL) переменного размера на место храненя этого сертификата в представлении DER. Это повышает эффективность в случаях кэширования данных сертификатов в конечных точках и делает IKE менее подверженным DoS-атакам, которые стали бы проще для больших сообщений IKE требующих фрагментации IP [DOSUDPPROT].

Тип «Хэш и URL связки» (Hash and URL of a bundle) использует определение ASN.1 для связки X.509 приведенное ниже.

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

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

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

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

```
   CertificateBundle ::= SEQUENCE OF CertificateOrCRL

   END
```

Реализации **должны** обеспечивать возможность настройки передачи и восприятия до четырех 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.

Реализации **должны** поддерживать метод HTTP для поиска hash-and-URL. Поведение других URL-методов в настоящее время не задается и такие методы **не следует** применять при отсутствии спецификаций для них.

3.7. Элемент Certificate Request

Элементе данных Certificate Request (далее, CERTREQ) обеспечивает способ запроса предпочтительных сертификатов через IKE и может присутствовать в отклике IKE_SA_INIT[1] и/или запросе IKE_AUTH. Элементы Certificate Request **могут** включаться в обмен, где отправителю нужно получить сертификат получателя.

Рисунок 13. Формат элемента Certificate Request

Поля Certificate Request описаны ниже.

Certificate Encoding (1 октет)

Указывает тип или формат запрашиваемого сертификата. Возможные значения указаны в параграфе 3.6.

Certification Authority (переменный размер)

Указывает представление приемлемого удостоверяющего центра для запрошенного типа сертификата.

Идентификатор типа для элемента Certificate Request имеет значение 38.

В поле Certificate Encoding используются те же значения, которые были определены в параграфе 3.6. Поле Certification Authority содержит индикатор доверенных УЦ для данного типа сертификата. Значение Certification Authority представляет собой конкатенацию хэш-значений SHA-1 для открытых ключей удостоверяющих центров (CA[2]). Каждое значение представляет собой хэш SHA-1 для элемента Subject Public Key Info (см. параграф 4.1.2.7 работы [PKIX]) из каждого сертификата Trust Anchor. Двадцатиоктетные хэш-значения объединяются (конкатенация) и помещаются в поле без дополнительного форматирования.

Содержимое поля Certification Authority определено только для сертификатов X.509 (типы 4, 12 и 13). Использовать другие значения **не следует**, пока не будут опубликованы соответствующие спецификации их применения со статусом Standards-Track.

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

[1] В оригинале ошибочно сказано IKE_INIT_SA, см. https://www.rfc-editor.org/errata_search.php?eid=4387. *Прим. перев.*
[2] Certification Authority — агентство по сертификации, удостоверяющий центр.

Уведомления HTTP_CERT_LOOKUP_SUPPORTED **могут** включаться в любые сообщения, которые могут включать элемент CERTREQ, для указания возможности отправителя просматривать сертификаты но ссылкам HTTP (URL) и, следовательно, его вероятного предпочтения сертификатов в таком формате.

3.8. Элемент Authentication

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

Поля элемента Authentication описаны ниже.

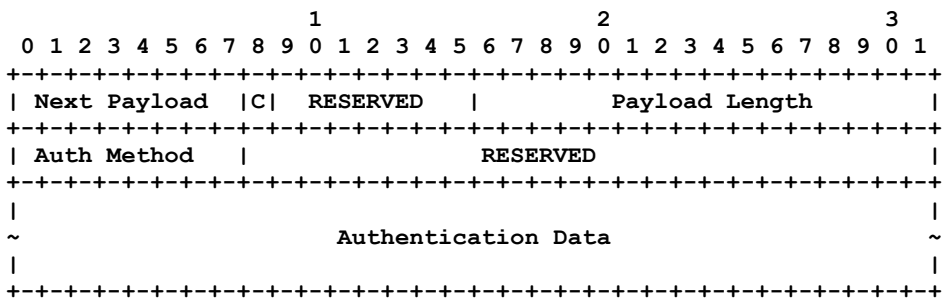


Рисунок 14. Формат элемента Authentication

Auth Method (1 октет)

Задает используемый метод аутентификации. Ниже перечислены типы подписей, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

RSA Digital Signature 1

Рассчитывается в соответствии с параграфом 2.15 с использованием секретного ключа RSA со схемой подписи RSASSA-PKCS1-v1_5, описанной в [PKCS1] (разработчикам следует принимать во внимание, что IKEv1 использует другой метод для подписей RSA). Для обеспечения взаимодействия реализациям, поддерживающим этот тип, **следует** поддерживать подписи, использующие SHA-1 в качестве хэш-функции, а также **следует** применять функцию SHA-1 по умолчанию при генерации подписей. Реализации могут использовать полученный от партнера сертификат, как рекомендацию по выбору понятной обеим сторонам хэш-функции для подписи в элементе AUTH. Однако следует отметить, что используемый в подписи элемента AUTH алгоритм хэширования не обязан совпадать с алгоритмами хэширования в сертификатах.

Shared Key Message Integrity Code 2

Рассчитывается в соответствии с параграфом 2.15 с использованием общего ключа, связанного с отождествлением в элементе ID, и согласованной функции PRF.

DSS Digital Signature 3

Рассчитывается в соответствии с параграфом 2.15 с использованием секретного ключа DSS (см. [DSS]) и хэширования SHA-1.

RESERVED

Должно устанавливаться в 0 при передаче и **должно** игнорироваться при получении.

Authentication Data (переменный размер)

См. параграф 2.15.

Идентификатор типа для элемента Authentication имеет значение 39.

3.9. Элемент 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.

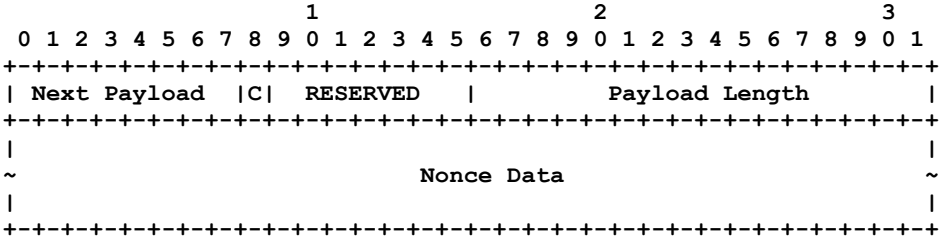


Рисунок 15. Формат элемента Nonce

Элемент Nonce включает единственное поле.

Nonce Data (переменный размер)

Случайные данные, генерируемые передающей стороной.

Идентификатор типа для элемента Nonce имеет значение 40.

Размер поля Nonce Data **должен** быть в диапазоне 16 - 256 октетов, включительно. **Недопустимо** повторное использование значений Nonce.

3.10. Элемент Notify

Элемент Notify (N) используется для передачи служебной информации (ошибки, смена состояния) партнеру 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.

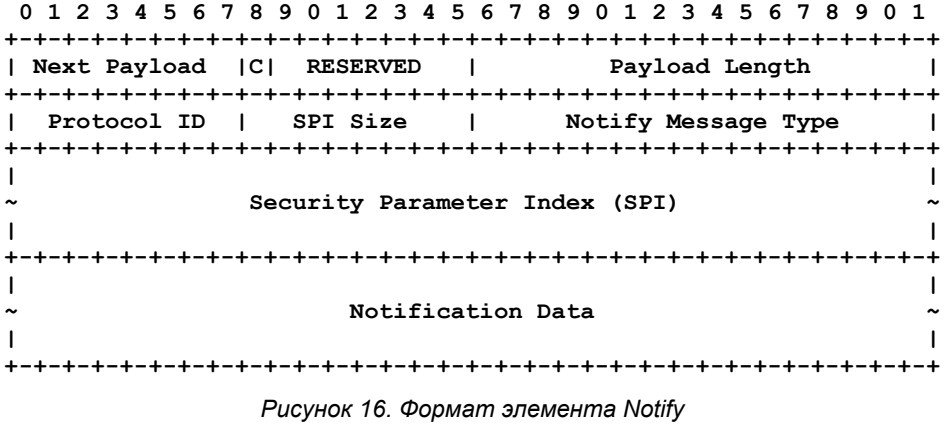


Рисунок 16. Формат элемента Notify

Поля элемента Notify описаны ниже.

Protocol ID (1 октет)
Если уведомление относится к существующей SA, значение SPI которой указано в поле SPI, это поле показывает тип данной SA. Для уведомлений, относящихся к Child SA, это поле **должно** содержать значение 2 для индикации протокола AH или 3 для индикации ESP. Из уведомлений, определенных в этом документе, поле SPI включается только в INVALID_SELECTORS, REKEY_SA и CHILD_SA_NOT_FOUND[1]. Если поле SPI пусто, в поле идентификатора протокола при передаче **должно** помещаться значение 0, а на приемной стороне это поле **должно** игнорироваться.

SPI Size (1 октет)
Размер SPI (в октетах) в соответствии с идентификатором протокола IPsec или 0, если индекс SPI не применим. Для уведомлений, касающихся IKE_SA, поле SPI Size **должно** иметь значение 0, а поле SPI должно быть пустым.

Notify Message Type (2 октета)
Указывает тип уведомления.

SPI (переменный размер)
Индекс параметров защиты.

Notification Data (переменный размер)
Данные о статусе или ошибке, передаваемые в дополнение к Notify Message Type. Значения этого поля определяются типом уведомления (см. ниже).

Идентификатор типа для элемента Notify имеет значение 41.

3.10.1. Типы сообщений Notify

Уведомления могут быть сообщениями об ошибках, указывающими причины отказов при организации SA. Они могут также содержать данные о состоянии, которыми процесс управления базой данных SA желает обмениваться с процессом партнера.

Ниже приведен список сообщений Notification с их кодами. Число разных сообщений об ошибках существенно снижено по сравнению с IKEv1 в целях упрощения и сокращения объема конфигурационных данных, доступных для зондирования.

Диапазон типов от 0 до 16383 предназначен для сообщений об ошибках. Реализация, получившая элемент Notify с кодом из этого диапазона, который она не распознает, при отклике **должна** предположить полный отказ при обработке соответствующего запроса. Непонятные типы ошибок в запросах и типы состояний в запросах и откликах **должны** игнорироваться с записью таких фактов в системные журналы.

Элементы Notify типов, относящихся к состояниям, **могут** добавляться в любые сообщения. Непонятные элементы таких типов **должны** игнорироваться. Такие сообщения предназначены для индикации возможностей и, как часть согласования SA, применяются для согласования некриптографических параметров.

Дополнительная информация по обработке ошибок приведена в параграфе 2.21.

Ниже приведены типы сообщений, определенные на момент публикации RFC 4306, а также два дополнительных типа, определенные в этом документе. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Типы ошибок для сообщений NOTIFY.

UNSUPPORTED_CRITICAL_PAYLOAD 1
См. параграф 2.5.

INVALID_IKE_SPI 4
См. параграф 2.21.

INVALID_MAJOR_VERSION 5
См. параграф 2.5.

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
См. параграф 2.3.

INVALID_SPI 11
См. параграф 1.5.

NO_PROPOSAL_CHOSEN 14
Ни один из предложенных шифронаборов не может быть принят. Такое сообщение может передаваться в любой ситуации, когда предложения (включая значения элементов SA, уведомления USE_TRANSPORT_MODE и IPCOMP_SUPPORTED и др.) не могут быть приняты ответчиком. Сообщение может также указывать ошибку общего типа для Child SA, когда связь Child SA не может быть создана по той или иной причине. См. также параграф 2.7.

INVALID_KE_PAYLOAD 17
См. параграфы 1.2 и 1.3.

AUTHENTICATION_FAILED 24
Передается в ответ на сообщение IKE_AUTH когда аутентификация по той или иной причине завершилась отказом. См. также параграф 2.21.2.

SINGLE_PAIR_REQUIRED 34
См. параграф 2.9.

[1] В оригинале это предложение содержит ошибку. См. http://www.rfc-editor.org/errata_search.php?eid=3036. *Прим. перев.*

NO_ADDITIONAL_SAS ***35***
См. параграф 1.3.
INTERNAL_ADDRESS_FAILURE ***36***
См. параграф 3.15.4.
FAILED_CP_REQUIRED ***37***
См. параграф 2.19.
TS_UNACCEPTABLE ***38***
См. параграф 2.9.
INVALID_SELECTORS ***39***
Может передаваться в обмене IKE INFORMATIONAL, когда узел получает пакет ESP или AH с селекторами, не соответствующими SA, использованной для его доставки (это требует отбрасывания пакета). Поле Notification Data указывает начало пакета-нарушителя (как в сообщениях ICMP), а поле SPI в уведомлении устанавливается в соответствии с SPI для Child SA.
TEMPORARY_FAILURE ***43***
См. параграф 2.25.
CHILD_SA_NOT_FOUND ***44***
См. параграф 2.25.

Типы состояний для сообщений NOTIFY.

INITIAL_CONTACT ***16384***
См. параграф 2.4.
SET_WINDOW_SIZE ***16385***
См. параграф 2.3.
ADDITIONAL_TS_POSSIBLE ***16386***
См. параграф 2.9.
IPCOMP_SUPPORTED ***16387***
См. параграф 2.22.
NAT_DETECTION_SOURCE_IP ***16388***
См. параграф 2.23.
NAT_DETECTION_DESTINATION_IP ***16389***
См. параграф 2.23.
COOKIE ***16390***
См. параграф 2.6.
USE_TRANSPORT_MODE ***16391***
См. параграф 1.3.1.
HTTP_CERT_LOOKUP_SUPPORTED ***16392***
См. параграф 3.6.
REKEY_SA ***16393***
См. параграф 1.3.3.
ESP_TFC_PADDING_NOT_SUPPORTED ***16394***
См. параграф 1.3.1.
NON_FIRST_FRAGMENTS_ALSO ***16395***
См. параграф 1.3.1.

3.11. Элемент Delete

Элемент Delete (далее, D) содержит определяемый протоколом идентификатор защищенной связи, которую отправитель удаляет из базы данных (следовательно, связь перестает 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.

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.

Поля элемента Delete описаны ниже.

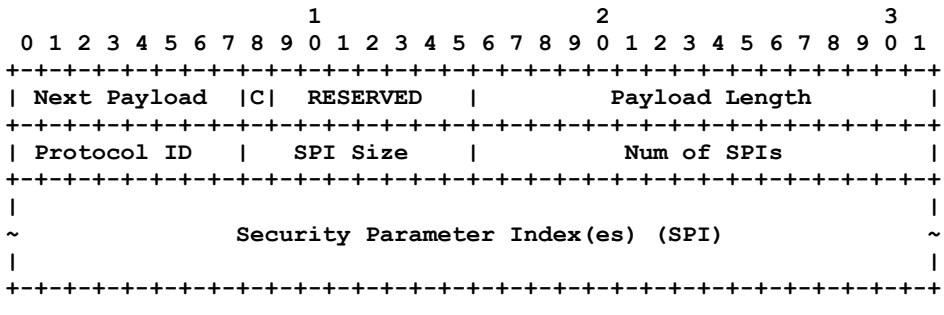


Рисунок 17. Формат элемента Delete

Protocol ID (1 октет)
1 для IKE SA, 2 для AH, 3 для ESP.
SPI Size (1 октет)
Размер (в октетах) индекса SPI, определяемого идентификатором протокола. Это поле **должно** иметь значение 0 для IKE (SPI заголовке сообщения) или 4 для AH и ESP.
Num of SPIs (2 октета, целое число без знака)
Число индексов SPI в элементе Delete. Размер отдельных SPI определяется полем SPI Size.
Security Parameter Index(es) (переменный размер)
Указывает конкретную удаляемую связь или связи SA. Размер этого поля определяется значениями полей SPI Size и Num of SPIs.

Идентификатор типа для элемента Delete имеет значение 42.

3.12. Элемент Vendor ID

Элемент данных Vendor ID (далее, V) содержит определенные производителем константы, которые служат для идентификации и распознавания удаленных 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 или просто идентифицировать реализацию в целях отладки. На основе значения **недопустимо** изменять интерпретацию какой-либо информации, определенной в этом документе (т. е. бит критичности **должен** быть сброшен). **Возможна** передача множества элементов Vendor ID. Реализации **не обязана** передавать элементы Vendor ID и может не использовать их совсем.

Элемент Vendor ID может передаваться в любом сообщении. Получение знакомого элемента Vendor ID дает 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 документов, желающие расширить данный протокол, **должны** определить элемент Vendor ID для анонсирования возможности реализовать расширение описанное в документе. Предполагается, что получившие признание документы, будут стандартизоваться с выделением значений из резервных блоков IANA (Future Use) и использование данного Vendor ID будет прекращаться.

Поля элемента Vendor ID описаны ниже.

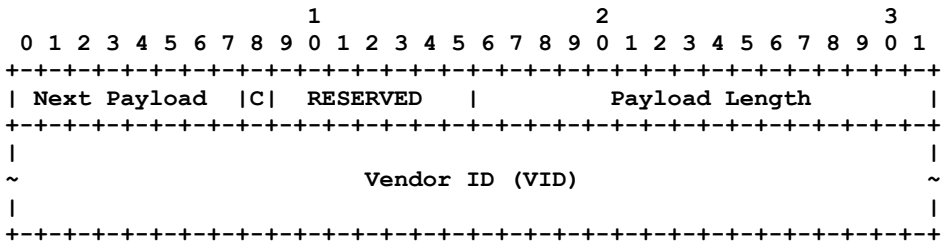


Рисунок 18. Формат элемента Vendor ID

Vendor ID (переменный размер)

Несмотря на отсутствие централизованного 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 для элемента Vendor ID имеет значение 43.

3.13. Элемент TS

Элемент данных Traffic Selector (TS) позволяет партнерам идентифицировать потоки пакетов для обработки службами IPsec. Элемент TS состоит из базового заголовка IKE, за которым следуют отдельные селекторы трафика.

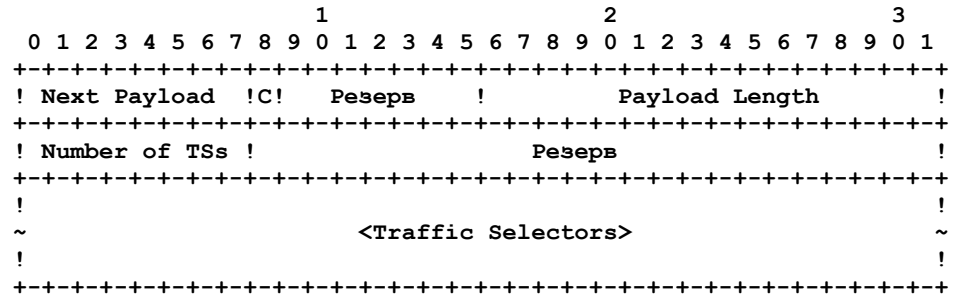


Рисунок 19: Формат элемента данных Traffic Selector

Number of TSs (1 октет)

Число представляемых селекторов трафика.

RESERVED

Должно устанавливаться в 0 при передаче и игнорироваться при получении.

Traffic Selectors (переменный размер)

Один или множество индивидуальных селекторов трафика.

Размер элемента TS включает заголовок TS и все селекторы трафика.

Идентификатор типа элемента Traffic Selector имеет значение 44 для адресов на стороне инициатора SA и 45 на стороне ответчика.

Элементы TSi и TSr не обязаны включать одинаковое число селекторов трафика. Пакет считается соответствующим данному Tsi/TSr, если он соответствует хотя бы одному индивидуальному селектору в TSi и хотя бы одному индивидуальному селектору в TSr.

Например, селекторам

```
TSi = ((17, 100, 198.51.100.66-198.51.100.66), (17, 200, 198.51.100.66-198.51.100.66))
TSr = ((17, 300, 0.0.0.0-255.255.255.255), (17, 400, 0.0.0.0-255.255.255.255))
```

будут соответствовать пакеты UDP с адреса 198.51.100.66 в любой адрес, использующие любую из комбинаций портов (100,300), (100,400), (200,300), (200, 400).

Таким образом, для некоторых типов правил может потребоваться несколько пар Child SA. Например, правило, соответствующее только портам отправителя/получателя (100,300) и (200,400) в приведенном выше примере, не может быть согласовано при одной паре Child SA.

3.13.1. Селектор трафика

TS Type (1 октет)

Задает тип селектора трафика.

IP protocol ID (1 октет)

Значение, указывающее идентификатор связанного протокола IP (например, UDP, TCP, ICMP). Нулевое значение указывает, что с данным TS не связан какой-то протокол – SA может применяться для всех протоколов.

Selector Length (2 октета, целое число без знака)

Указывает размер данной субструктуры Traffic Selector с учетом заголовка.

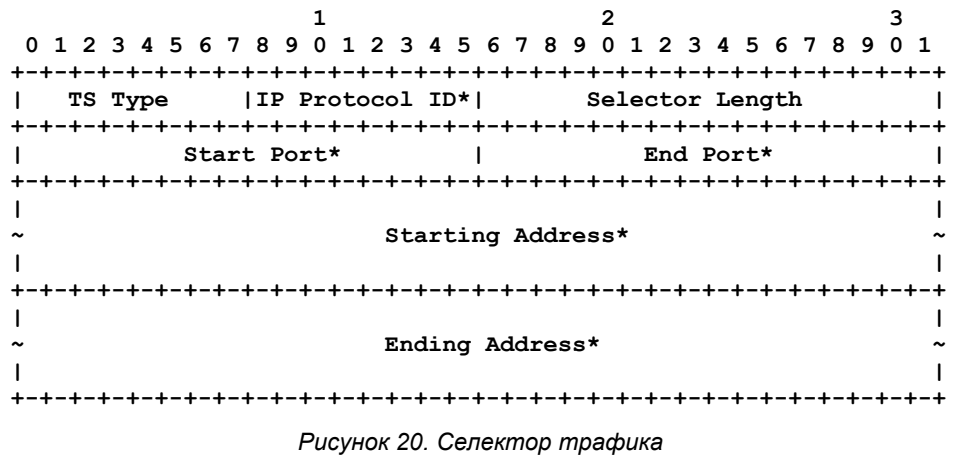


Рисунок 20. Селектор трафика

** (все поля, кроме TS Type и Selector Length зависят от TS Type; на рисунке приведены поля для типов 7 и 8, определенных в настоящее время)*

Start Port (2 октета, целое число без знака)

Наименьший номер порта, разрешенный для этого селектора. Для протоколов, где номер порта не определен (включая protocol 0) или разрешены любые номера, это поле **должно** иметь значение 0. Значения кода и типа ICMP и ICMPv6, а также типы заголовка (MH) Mobile IP версии 6 (MIPv6) представляются в этом поле в соответствии с параграфом 4.4.1.1 [IPSECARCH]. Значения типа и кода ICMP трактуются, как 16-битовый целочисленный номер порта, где старшие 8 битов указывают тип, а младшие 8 - код. Значения MIPv6 MH Type трактуются, как 16-битовый целочисленный номер порта, где старшие 8 битов указывают тип, а младшие 8 имеют значение 0.

End Port (2 октета, целое число без знака)

Наибольший номер порта, разрешенный для этого селектора. Для протоколов, где номер порта не определен (включая protocol 0) или разрешены любые номера, это поле **должно** иметь значение 65535. Значения кода и типа ICMP и ICMPv6, а также типы заголовка (MH) Mobile IP версии 6 (MIPv6) представляются в этом поле в соответствии с параграфом 4.4.1.1 [IPSECARCH]. Значения типа и кода ICMP трактуются, как 16-битовый целочисленный номер порта, где старшие 8 битов указывают тип, а младшие 8 - код. Значения MIPv6 MH Type трактуются, как 16-битовый целочисленный номер порта, где старшие 8 битов указывают тип, а младшие 8 имеют значение 0.

Starting Address

Наименьший адрес, включенный в этот селектор трафика (размер определяется полем TS Type).

Ending Address

Наибольший адрес, включенный в этот селектор трафика (размер определяется полем TS Type).

Системы, соответствующие [IPSECARCH] 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 [IPSECARCH] и желающие показать порты OPAQUE, но не ANY, **должны** установить для начального порта значение 65535, а для конечного - 0.

Селекторы трафика типов 7 и 8 могут также указывать поля типа и кода ICMP или ICMPv6, равно как тип полей MH Type в заголовках IPv6 mobility [MIPV6]. Однако следует отметить, что пакеты ICMP и MIPv6 не имеют раздельных полей для отправителя и получателя. Метод задания селекторов трафика для ICMP и MIPv6 показан в примере параграфа 4.4.1.3 [IPSECARCH].

Ниже приведены значения поля Traffic Selector Type с описаниями адресных селекторов, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

TS_IPV4_ADDR_RANGE 7

Диапазон адресов IPv4, представленный двумя 4-октетными значениями. Первое значение указывает начальный, в второе - конечный адрес IPv4 (оба значения включаются в диапазон адресов, вместе со всеми адресами внутри диапазона).

TS_IPV6_ADDR_RANGE 8

Диапазон адресов IPv6, представленный двумя 16-октетными значениями. Первое значение указывает начальный, в второе - конечный адрес IPv6 (оба значения включаются в диапазон адресов, вместе со всеми адресами внутри диапазона).

3.14. Элемент Encrypted

Элемент Encrypted, обозначаемый в этом документе SK{...}, содержит другие элементы в зашифрованном виде. При наличии элемента Encrypted этот элемент **должен** быть последним в сообщении. Часто этот элемент является единственным элементом данных в сообщении. Этот элемент называют также Encrypted and Authenticated (зашифровано и аутентифицировано).

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

В этом документе описана криптографическая обработка элементов Encrypted с использованием блочного шифра в режиме CBC и алгоритма защиты целостности с контрольными суммами фиксированного размера, рассчитываемыми для сообщений переменного размера. Устройство протокола разрабатывалось 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. В будущих документах может быть задана обработка элементов Encrypted для других видов преобразований типа алгоритмов шифрования со счетчиком (counter mode encryption) и аутентифицированного шифрования (authenticated encryption). Партнерам **недопустимо** согласовывать преобразования, для которых нет спецификаций.

При использовании алгоритма аутентифицированного шифрования для защиты IKE SA устройство элементов Encrypted отличается от описанного здесь. В документе [AEAD] представлена информация о таких алгоритмах и их использовании в IKEv2.

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

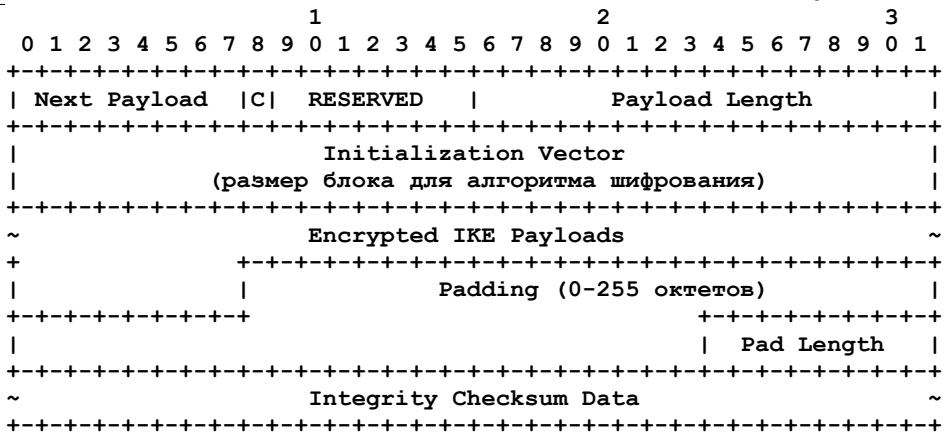


Рисунок 21. Формат элемента Encrypted

Next Payload

Указывает тип первого вложенного элемента данных. Отметим, что это отличается от стандартного формата заголовка, поскольку элемент Encrypted является последним в сообщении и в этом случае полю Next Payload следовало бы иметь значение 0. Однако по причине того, что содержимым этого элемента данных являются другие (вложенные) элементы и отсутствует логичное место для размещения информации о типе первого элемента, этот тип помещен сюда.

Payload Length

Размер элемента данных с учетом полей вектора инициализации (IV), Encrypted IKE Payloads, Padding, Pad Length, и Integrity Checksum Data.

Initialization Vector

Для шифров в режиме CBC размер вектора инициализации (IV) совпадает с размером блока алгоритма шифрования. Отправитель **должен** выбирать новый непредсказуемый вектор IV для каждого сообщения, получатели **должны** воспринимать любое значение. Читателям рекомендуется обратиться к работе [MODES] по вопросам генерации IV. В частности, использование последнего зашифрованного блока из предыдущего сообщения не считается непредсказуемым. Для отличных от CBC режимов формат и обработка IV задается в документах, определяющих алгоритм шифрования и режим работы.

IKE payloads

Соответствует приведенному выше описанию. Это поле шифруется с использованием согласованного алгоритма.

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. Элемент Configuration

Элемент данных Configuration (CP) используется для обмена конфигурационными параметрами между партнерами IKE. Такой обмен используется для запроса клиентом IRAC внутреннего адреса IP у сервера IRAS, а также для обмена другой информацией в процессе получения адреса по протоколу DHCP[1], если клиент IRAC подключен непосредственно к ЛВС.

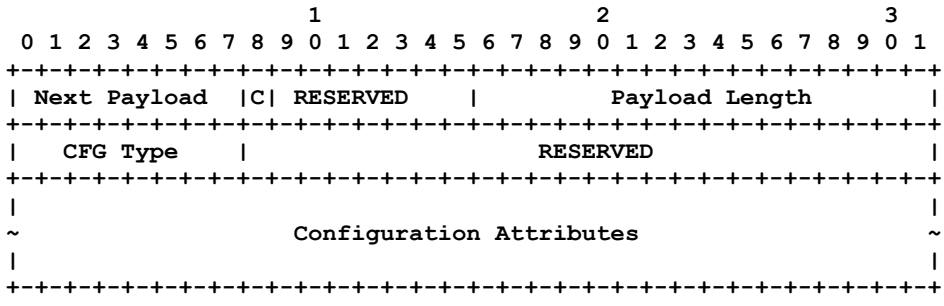


Рисунок 22. Формат элемента Configuration

Поля элемента Configuration описаны ниже.

Идентификатор типа для элемента Configuration имеет значение 47.

CFG Type (1 октет)

Тип обмена, представленного полем Configuration Attributes. В таблице приведены идентификаторы типов, определенные на момент публикации RFC 4306. Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Тип	Значение
CFG_REQUEST	1
CFG_REPLY	2
CFG_SET	3
CFG_ACK	4

RESERVED (3 октета)

Должно устанавливаться в 0 при передаче и игнорироваться при получении.

[1]Dynamic Host Configuration Protocol — протокол динамической настройки конфигурации хоста.

Configuration Attributes (переменный размер)

Структуры TLV[1] конкртеного элемента Configuration, описанные ниже. Элемент может включать множество атрибутов или не иметь их совсем.

3.15.1. Атрибуты конфигурации

Reserved (1 бит)

Должно устанавливаться в 0 при передаче и игнорироваться при получении.

Attribute Type (15 битов)

Уникальный идентификатор типа конфигурационного атрибута.

Length (2 октета, целое число без знака)

Размер значения в октетах.

Value (0 или более октетов)

Значение атрибута (переменный размер). Список возможных типов атрибутов приведен ниже.

Рисунок 23. Формат атрибута Configuration

В таблице приведен список атрибутов конфигурации, определенных на момент публикации RFC 4306 (за исключением INTERNAL_ADDRESS_EXPIRY и INTERNAL_IP6_NBNS, которое отменены RFC 5996). Новые значения могли быть добавлены с тех пор и могут быть добавлены после публикации этого документа. Актуальные значения следует смотреть в [IKEV2IANA].

Тип атрибута	Значение	Многозначный	Размер
INTERNAL_IP4_ADDRESS	1	да*	0 или 4 октета
INTERNAL_IP4_NETMASK	2	нет	0 или 4 октета
INTERNAL_IP4_DNS	3	да	0 или 4 октета
INTERNAL_IP4_NBNS	4	да	0 или 4 октета
INTERNAL_IP4_DHCP	6	да	0 или 4 октета
APPLICATION_VERSION	7	нет	0 или более
INTERNAL_IP6_ADDRESS	8	да*	0 или 17 октетов
INTERNAL_IP6_DNS	10	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

**) Эти атрибуты могут быть многозначными при возврате только в тех случаях, когда запрошена многозначность.*

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. В запросе указанный адрес является запрашиваемым (или адресом нулевого размера, если конкретный 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 остается корректным в течение всего срока действия IKE SA (или ее наследников после смены ключей). Этот вопрос более подробно рассмотрен в параграфе 3.15.3.

INTERNAL_IP4_NETMASK

Маска внутренней сети. В запросах и откликах допускается только одна маска (например, 255.255.255.0) и она **должна** использоваться только с атрибутом INTERNAL_IP4_ADDRESS. Атрибут INTERNAL_IP4_NETMASK в CFG_REPLY означает то же самое, что атрибут INTERNAL_IP4_SUBNET, содержащий ту же информацию (передавать трафик для этих адресов через меня), но подразумевает границу сети. Например, клиент может использовать свой собственный адрес и маску при расчете широковещательного адреса для канала. Пустой атрибут INTERNAL_IP4_NETMASK может включаться в CFG_REQUEST для запроса информации (хотя шлюз может передать информацию даже без запроса). Непустые значения этого атрибута в CFG_REQUEST не имеют смысла и их включение **недопустимо**.

INTERNAL_IP4_DNS, INTERNAL_IP6_DNS

Указывает адрес сервера DNS в сети. Запрашивать **можно** множество серверов DNS. Ответчик **может** вернуть адреса множества серверов или не вернуть ни одного.

INTERNAL_IP4_NBNS

Указывает адрес сервера сервера имен NetBios (WINS) в сети. Запрашивать **можно** множество серверов NBNS. Ответчик **может** вернуть адреса множества серверов или не вернуть ни одного.

INTERNAL_IP4_DHCP, INTERNAL_IP6_DHCP

Рекомендует хосту передавать внутренние запросы DHCP по адресу, указанному в этом атрибуте. Запрашивать **можно** множество серверов DHCP. Ответчик **может** вернуть адреса множества серверов или не вернуть ни одного.

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 задает адрес IP, а второе - маску. **Можно** запрашивать множество подсетей. Ответчик **может** вернуть множество подсетей или не вернуть ни одной. Дополнительное рассмотрение этого вопроса приведено в параграфе 3.15.2.

[1]Type-length-value — тип, размер, значение.

SUPPORTED_ATTRIBUTES

При использовании в запросе этот атрибут **должен** иметь нулевой размер и указывает ответчику запрос на возврат списка всех поддерживаемых тем атрибутов. В откликах этот атрибут содержит набор 2-октетных идентификаторов атрибутов. Размер атрибута, разделенный на 2, будет указывать число поддерживаемых атрибутов в отклике.

INTERNAL_IP6_SUBNET

Защищаемые данным краевым устройством подсети. Атрибут может содержать до двух полей, первое из которых задает 16-октетный адрес IPv6, а второе - маску. **Можно** запрашивать множество подсетей. Ответчик **может** вернуть множество подсетей или не вернуть ни одной. Дополнительное рассмотрение этого вопроса приведено в параграфе 3.15.2.

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

Пара CFG_REQUEST и CFG_REPLY позволяет конечной точке IKE запросить информацию у своего партнера. Если размер атрибута в элементе Configuration сообщения CFG_REQUEST не равен 0, такой элемент воспринимается как предложение атрибута. В элементе Configuration сообщения CFG_REPLY CFG_REPLY **может** быть возвращено это или новое значение атрибута. В отклик **могут** также добавляться атрибуты, не включенные в запрос. Непонятные или неподдерживаемые атрибуты **должны** игнорироваться в запросах и откликах.

Пара CFG_SET и CFG_ACK позволяет конечной точке передать (втолкнуть — push) конфигурационные данные своему партнеру. В этом случае элемент Configuration сообщения CFG_SET указывает атрибуты, которые инициатор желает изменить у партнера. Ответчик **должен** возвращать элемент Configuration payload, если он воспринял какие-либо из предложенных конфигурационных данных, и **должен** указывать воспринятые атрибуты с нулевым размером данных. Не воспринятые атрибуты **недопустимо** включать в элемент Configuration сообщения CFG_ACK. Если не воспринят ни один из предложенных атрибутов, ответчик **должен** вернуть пустой элемент CFG_ACK или отклик без элемента CFG_ACK. В настоящее время использование обмена CFG_SET/CFG_ACK не определено, хотя он может применяться в комбинациях с расширениями на базе Vendor ID. Реализации **могут** игнорировать элементы CFG_SET.

3.15.2. Значение INTERNAL_IP4_SUBNET и INTERNAL_IP6_SUBNET

Атрибуты INTERNAL_IP4/6_SUBNET млгут указывать дополнительные подсети, для которых может потребоваться одна или множество отдельных связей SA, доступных через шлюз, анонсируемый этими атрибутами. Атрибуты INTERNAL_IP4/6_SUBNET могут также выражать политику шлюза для трафика, который следует передавать через него - клиент может выбрать передачу другого трафика (входящего в TSr, но не в INTERNAL_IP4/6_SUBNET) через шлюз или напрямую адресату. Таким образом, трафик по адресам, указанным атрибутами INTERNAL_IP4/6_SUBNET, следует передавать через анонсируемые этими атрибутами шлюзы. Если нет Child SA, чьи селекторы трафика включают рассматриваемый адрес, требуется создание новых связей SA.

Например, если имеются две подсети 198.51.100.0/26 и 192.0.2.0/24, а запрос клиента включает

```
CP(CFG_REQUEST) = INTERNAL_IP4_ADDRESS()
TSi = (0, 0-65535, 0.0.0.0-255.255.255.255)
TSr = (0, 0-65535, 0.0.0.0-255.255.255.255)
```

приемлемый отклик может иметь вид (TSr и INTERNAL_IP4_SUBNET содержат одинаковую информацию)

```
CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_SUBNET(198.51.100.0/255.255.255.192)
  INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0)
TSi = (0, 0-65535, 198.51.100.234-198.51.100.234)
TSr = ((0, 0-65535, 198.51.100.0-198.51.100.63),
       (0, 0-65535, 192.0.2.0-192.0.2.255))
```

В таких случаях INTERNAL_IP4_SUBNET реально не содержит какой-либо полезной информации.

Другой возможный отклик имеет вид

```
CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_SUBNET(198.51.100.0/255.255.255.192)
  INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0)
TSi = (0, 0-65535, 198.51.100.234-198.51.100.234)
TSr = (0, 0-65535, 0.0.0.0-255.255.255.255)
```

Этот отклик будет означать, что клиент может передавать весь свой трафик через шлюз, но этот шлюз не будет возражать, если клиент станет передавать трафик, не включенный в INTERNAL_IP4_SUBNET напрямую адресатам (без прохождения через шлюз).

Другая ситуация возникает, если на шлюзе имеется правило, требующее передавать трафик для двух подсетей через разные SA. Отклик, показанный ниже, будет говорить клиенту о необходимости создания отдельной связи SA для доступа во вторую подсеть.

```
CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_SUBNET(198.51.100.0/255.255.255.192)
  INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0)
TSi = (0, 0-65535, 198.51.100.234-198.51.100.234)
TSr = (0, 0-65535, 198.51.100.0-198.51.100.63)
```

Атрибут INTERNAL_IP4_SUBNET может быть также полезен в случаях, когда клиентский селектор TSr включает лишь часть адресного пространства. Например, если клиент запрашивает

```
CP(CFG_REQUEST) =
  INTERNAL_IP4_ADDRESS()
```

```
   TSi = (0, 0-65535, 0.0.0.0-255.255.255.255)
   TSr = (0, 0-65535, 192.0.2.155-192.0.2.155)
```

отклик шлюза может иметь вид

```
   CP(CFG_REPLY) =
     INTERNAL_IP4_ADDRESS(198.51.100.234)
     INTERNAL_IP4_SUBNET(198.51.100.0/255.255.255.192)
     INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0)
   TSi = (0, 0-65535, 198.51.100.234-198.51.100.234)
   TSr = (0, 0-65535, 192.0.2.155-192.0.2.155)
```

Поскольку смысл атрибутов INTERNAL_IP4_SUBNET/INTERNAL_IP6_SUBNET в сообщениях CFG_REQUEST не ясен, их невозможно надежно применять в CFG_REQUEST.

3.15.3. Элемент Configuration для IPv6

Элементы Configuration для IPv6, основанные на соответствующих элементах для IPv4, не вполне соответствуют «обычному стилю» IPv6. В частности, не используется автонастройка IPv6 без учета состояний, сообщения с анонсами маршрутизаторов и обнаружение соседей. Отметим наличие дополнительного документа, посвященного конфигурации IPv6 в IKEv2 [IPV6CONFIG]. В настоящее время это экспериментальный документ, но есть надежда по результатам экспериментов сделать его стандартом.

Клиенту можно выделить адрес IPv6, используя атрибут INTERNAL_IP6_ADDRESS элемента Configuration. Минимальный обмен при этом может иметь вид

```
   CP(CFG_REQUEST) =
     INTERNAL_IP6_ADDRESS()
     INTERNAL_IP6_DNS()
   TSi = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)
   TSr = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)

   CP(CFG_REPLY) =
     INTERNAL_IP6_ADDRESS(2001:DB8:0:1:2:3:4:5/64)
     INTERNAL_IP6_DNS(2001:DB8:99:88:77:66:55:44)
   TSi = (0, 0-65535, 2001:DB8:0:1:2:3:4:5 - 2001:DB8:0:1:2:3:4:5)
   TSr = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)
```

Клиент **может** передать непустой атрибут INTERNAL_IP6_ADDRESS в CFG_REQUEST для запроса конкретного адреса или идентификатора интерфейса. Шлюз сначала проверяет доступность указанного адреса и, при наличии, возвращает этот адрес. Если адрес не доступен, шлюз пытается использовать идентификатор интерфейса с некоторым другим префиксом, а при отказе будет выбирать другой идентификатор интерфейса.

Атрибут INTERNAL_IP6_ADDRESS содержит также поле размера префикса. При использовании в CFG_REPLY это поле соответствует атрибуту INTERNAL_IP4_NETMASK для случая IPv4.

Хотя описанная модель настройки адресов IPv6 достаточно проста, ей присущи некоторые ограничения. Туннели IPsec, настроенные с использованием IKEv2 не являются полнофункциональными «интерфейсами» в понимании архитектуры адресации IPv6 [ADDRIPV6]. В частности, у него может не быть локального для канала адреса (link-local) и это может осложнять использование протоколов, предполагающих его наличие (например, [MLDV2]).

3.15.4. Отказы при выделении адресов

Если ответчик сталкивается с ошибкой при попытке выделить инициатору адрес IP в процессе обработки элемента Configuration, он отвечает уведомлением INTERNAL_ADDRESS_FAILURE. Связь IKE SA все равно создается, даже если упомянутый отказ не позволяет организовать начальную Child SA. Если ошибка возникает в обмене IKE_AUTH, связи Child SA не будут организованы. Однако имеются и более сложные ситуации.

Если ответчик совсем не поддерживает элементы Configuration, он может просто игнорировать их. Реализации такого типа никогда не передают уведомления INTERNAL_ADDRESS_FAILURE. Если инициатору требуется выделение адреса IP, он будет трактовать отклик CFG_REPLY, как ошибку.

Инициатор может запросить конкретный тип адреса (IPv4 или IPv6), который не поддерживается ответчиком, даже если ответчик поддерживает элементы данных Configuration. В таких случаях ответчик просто игнорирует не поддерживаемый тип адреса, выполняя обычную обработку оставшейся части запроса.

Если инициатор не получает адрес(а), требуемый его политикой, он **может** сохранить IKE SA и повторить элемент Configuration в отдельном обмене INFORMATIONAL после некоторого ожидания или **может** разорвать связь IKE SA, передав элемент Delete в отдельном обмене INFORMATIONAL, а позднее повторить попытку организации IKE SA. Время ожидания в описанных случаях не следует выбирать слишком малым (особенно при организации IKE SA заново), поскольку приведшие к ошибке ситуации могут сохраняться довольно долго. Разумным представляется интервал в несколько минут. Например, проблема нехватки адресов у ответчика может сохраняться, пока не будут возвращены адреса, используемые другими клиентами, или не будет изменена конфигурация ответчика с расширением доступного блока адресов.

3.16. Элемент EAP

Элемент Extensible Authentication Protocol[1] (EAP) позволяет выполнять аутентификацию IKE_SA с использованием протокола, определенного в RFC 3748 [EAP] и его последующих расширений. При использовании EAP требуется выбрать подходящий метод. Определено множество таких методов, описывающих использование протокола с разными механизмами аутентификации. Типы методов EAP перечислены в [EAP-IANA]. Здесь включено краткое описание формата EAP для ясности.

[1]Расширяемый протокол аутентификации.

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).

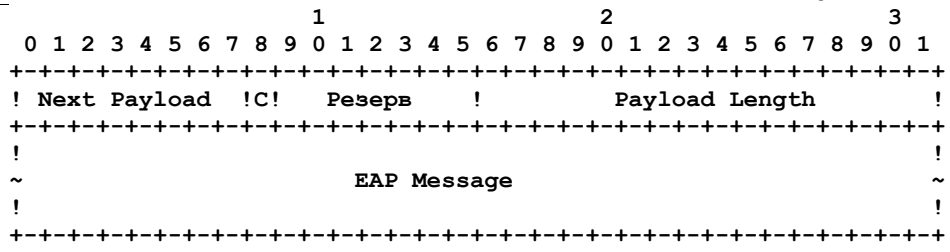


Рисунок 24/ Формат EAP

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.

Length (2 октета, целое число без знака)

Указывает размер сообщения EAP и **должно** иметь значение на 4 меньше значения Payload Length в инкапсулирующем элементе.

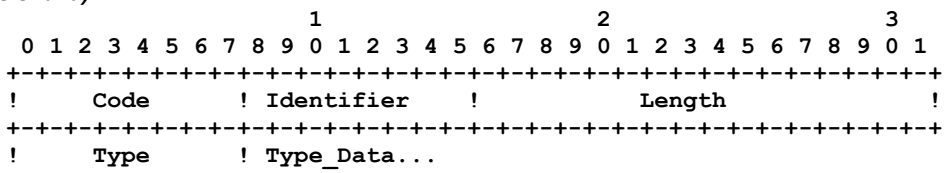


Рисунок 25. Формат сообщения EAP

Type (1 октет)

Присутствует только для элементов с кодом Request (1) или Response (2). Для других кодов размер сообщения EAP **должен** составлять четыре октета, а присутствие полей Type и Type_Data **недопустимо**. В запросах поле Type указывает запрашиваемые данные. В откликах в поле Type **должно** быть значение Nak или данные запрошенного типа. Поскольку протокол IKE передает индикацию отождествления инициатора в первом сообщении обмена IKE_AUTH, ответчику **не следует** передавать запросов EAP Identity (тип 1). Однако инициатор **может** отвечать на такие запросы при их получении.

Type_Data (переменный размер)

Зависит от типа запроса и связанного с ним отклика. Описание методов EAP приведено в документе [EAP].

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:

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

Для обеспечения интероперабельности 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 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 и политика требует использовать временный адрес IP, в первое сообщение обмена IKE_AUTH **должен** включаться 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** разобрать элемент CP типа CFG_REPLY в первом сообщении обмена IKE_AUTH, а также поле 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. Ответчик может включать любые другие относящиеся к делу атрибуты.

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

- инфраструктуры открытых ключей (PKI[1]), использующей сертификаты X.509 (PKIX), содержащие ключи RSA размером 1024 или 2048 и подписанные 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 применением сертификатов PKIX, а инициатора - с применением общего ключа.

5. Вопросы безопасности

Хотя протокол разработан так, чтобы минимизировать раскрытие конфигурационной информации неуполномоченным партнерам, некоторое раскрытие все-таки неизбежно. Один из партнеров 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. Разработчикам следует отметить этот факт и установить предел для числа обменов CREATE_CHILD_SA между сторонами. В этом документе данный предел не задается.

Стойкость ключей, созданных в результате обмена Diffie-Hellman с использованием любой из определенных 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 определенных 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.

Обмены IKE_SA_INIT и IKE_AUTH происходят до того, как аутентифицируется инициатор. В результате этого от реализаций данного протокола требуется полная устойчивость к развертыванию в любой незащищенной сети. Уязвимости реализаций, особенно для DoS-атак, могут быть использованы неуполномоченными партнерами. Эта проблема вызывает особое беспокойство с учетом неограниченного числа сообщений при аутентификации EAP.

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 значения должны создаваться качественным генератором случайных чисел или источником псевдослучайных чисел с подходящей «затравкой» (см. [RANDOMNESS]). 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 распространенных общих (pre-shared) ключей критически важным становится вопрос уровня хаотичности этих секретов. Жесткая практика требует обеспечения для этих ключей уровня хаотичности не меньше, чем у самого стойкого из согласуемых ключей. Создание общих ключей из паролей, имен 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

[1]Public Key Infrastructure.

выделить из пакетов. Это снижает число попыток расчета хэш-значений до 2^{32}. Когда можно обоснованно предположить использование адресов из приватных блоков[1], объем расчетов дополнительно сокращается во много раз. Разработчикам, в результате, не следует полагаться на то, что использование IKE гарантирует отсутствие утечки внутренней адресной информации.

При использовании методов аутентификации EAP без генерации общих ключей для защиты последующих элементов данных AUTH становятся возможными некоторые MITM[2]-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[3], развернутое решение IPsec, которое опирается на аутентификацию EAP без генерации общего ключа (этот метод также называют EAP без генерации ключей), может быть скомпрометировано в результате развертывания 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 защищенных 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 развертывании 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 путем опустошения ресурсов (заполнения буферов) для сборки фрагментов. Вероятность такого исхода можно минимизировать за счет использования кодирования «Hash and URL» вместо передачи сертификатов (см. параграф 3.6). Дополнительные меры по снижению рисков обсуждаются в работе [DOSUDPPROT].

Контроль доступа имеет важное значение для безопасности протокола. Например, доверенные привязки, используемые для идентификации партнеров IKE, следует, по возможности, отделять от привязок, используемых для идентификации общедоступных web-серверов. Протокол IKE обеспечивает значительную свободу при выборе политики безопасности в части идентификации доверенных партнеров и их свидетельств (credentials), а также связей между ними и наличие политики безопасности с явными определениями в этой части весьма важно для реализации системы защиты.

5.1. Полномочия для селекторов трафика

IKEv2 основывается на данных PAD[4] при определении типов Child SA, которые разрешено создавать партнеру. Этот процесс описан в параграфе 4.4.3 [IPSECARCH]. Когда партнер запрашивает создание Child SA с некими селекторами трафика, PAD должна содержать запись Child SA Authorization Data (данные о полномочиях Child SA) связывающую отождествление партнера, аутентифицированное IKEv2, и адреса, разрешенные для селекторов трафика.

Например, PAD может содержать запись, разрешающую партнеру с отождествлением sgw23.example.com создавать связи Child SA для адресов 192.0.2.0/24, которая означает, что данный защитный шлюз является приемлемым «представителем» для этих адресов. Для прямого взаимодействия между хостами IPsec требует наличия аналогичной записи. Например, fooserver4.example.com с 198.51.100.66/32 будет говорить о том, что указанное отождествление является приемлемым «владельцем» или «представителем» для данного адреса.

Как отмечено в [IPSECARCH], «Требуется вносить некоторые ограничения на создание дочерних SA, чтобы обезопасить идентифицированного партнера от обманных ID, связанных с другими легитимными партнерами.[5]» В приведенном выше примере корректная конфигурация PAD препятствует sgw23 в создании связей Child SA с адресом 198.51.100.66, а fooserver4 - в создании Child SA с адресами 192.0.2.0/24.

Важно отметить, сто простая передача пакетов IKEv2, использующих некий конкретный адрес, не подразумевает разрешения на создание связей Child SA с этим адресом в селекторах трафика. Например, даже если sgw23 сможет подменить свой адрес IP на 198.51.100.66, он не сможет создать связей Child SA, соответствующих селекторам трафика fooserver4.

Спецификация IKEv2 не задает точного взаимодействия процесса выделения адресов IP с использованием элементов Configuration и базы данных PAD. Наша интерпретация состоит в том, что при выделении защитным шлюзом адреса с использованием элементов данных Configuration этот шлюз создает также временную запись в PAD, связывающую отождествление партнера и вновь выделенный внутренний адрес.

Отмечено, что в некоторых средах корректная настройка конфигурации PAD может оказаться сложной. Например, если IPsec используется между парой хостов с динамическими адресами DHCP, очень сложно обеспечить указание в PAD корректного «владельца» для таких адресов IP. Это потребует защищенного механизма передачи сведений о выделении адресов от сервера DHCP и связывания этих хостов с отождествлениями, аутентифицированными с помощью IKEv2.

В силу таких ограничений некоторые производители настраивают конфигурацию своих PAD так, что аутентифицированным партнерам разрешается создавать связи Child SA с селекторами трафика, содержащими тот же адрес, который используется для пакетов IKEv2. В средах, где возможно использование обманных адресов IP (т. е., почти повсеместно) такой подход позволяет любому партнеру создавать Child SA с любыми селекторами трафика. Во

[1]См. RFC 1918. *Прим. перев.*
[2]Man-in-the-middle — атака на основе перехвата пакетов в пути с участием человека для анализа и изменения их. *Прим. перев.*
[3]Single-signon facility.
[4]Peer Authorization Database — база данных о полномочиях партнеров.
[5]Цитата приведена по опубликованному на сайте переводу. *Прим. перев.*

многих случаях такая конфигурация не будет ни подходящей, ни безопасной. Подробное обсуждение этого вопроса и общих ограничений для коммуникаций IPsec между парами хостов приведено в работе [H2HIPSEC].

6. Согласование с IANA

[IKEV2] определяет множество типов и значений полей. Агентство IANA уже внесло эти типы и значения в реестры [IKEV2IANA], поэтому они не указываются здесь.

Запись Raw RSA Key из реестра IKEv2 Certificate Encodings была признана устаревшей.

Агентство IANA заменило все ссылки на RFC 5996 ссылками на настоящий документ.

7. Литература

7.1. Нормативные документы

[ADDGROUP] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003, <http://www.rfc-editor.org/info/rfc3526>.

[ADDRIPV6] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[AEAD] Black, D. and D. McGrew, "Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol", RFC 5282, August 2008, <http://www.rfc-editor.org/info/rfc5282>.

[AESCMACPRF128] Song, J., Poovendran, R., Lee, J., and T. Iwata, "The Advanced Encryption Standard-Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-CMAC-PRF-128) Algorithm for the Internet Key Exchange Protocol (IKE)", RFC 4615, August 2006, <http://www.rfc-editor.org/info/rfc4615>.

[AESXCBCPRF128] Hoffman, P., "The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)", RFC 4434, February 2006, <http://www.rfc-editor.org/info/rfc4434>.

[EAP] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004, <http://www.rfc-editor.org/info/rfc3748>.

[ECN] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[ESPCBC] Pereira, R. and R. Adams, "The ESP CBC-Mode Cipher Algorithms", RFC 2451, November 1998, <http://www.rfc-editor.org/info/rfc2451>.

[IKEV2IANA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <http://www.iana.org/assignments/ikev2-parameters/>.

[IPSECARCH] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

[MUSTSHOULD] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[PKCS1] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003, <http://www.rfc-editor.org/info/rfc3447>.

[PKIX] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC4307] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", RFC 4307, December 2005, <http://www.rfc-editor.org/info/rfc4307>.

[UDPENCAPS] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005, <http://www.rfc-editor.org/info/rfc3948>.

[URLS] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

7.2. Дополнительная литература

[AH] Kent, S., "IP Authentication Header", RFC 4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[ARCHGUIDEPHIL] Bush, R. and D. Meyer, "Some Internet Architectural Guidelines and Philosophy", RFC 3439, December 2002, <http://www.rfc-editor.org/info/rfc3439>.

[ARCHPRINC] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996, <http://www.rfc-editor.org/info/rfc1958>.

[Clarif] Eronen, P. and P. Hoffman, "IKEv2 Clarifications and Implementation Guidelines", RFC 4718, October 2006, <http://www.rfc-editor.org/info/rfc4718>.

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

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

[DIFFSERVARCH] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998, <http://www.rfc-editor.org/info/rfc2475>.

[DIFFSERVFIELD] 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, <http://www.rfc-editor.org/info/rfc2474>.

[DIFFTUNNEL] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[DOI] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998, <http://www.rfc-editor.org/info/rfc2407>.

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

[DSS] National Institute of Standards and Technology, U.S. Department of Commerce, "Digital Signature Standard (DSS)", FIPS 186-4, July 2013, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[EAI] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, February 2012, <http://www.rfc-editor.org/info/rfc6532>.

[EAP-IANA] IANA, "Extensible Authentication Protocol (EAP) Registry: Method Types", <http://http://www.iana.org/assignments/eap-eke/>.

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

[ESP] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[EXCHANGEANALYSIS] Perlman, R. and C. Kaufman, "Analysis of the IPsec key exchange Standard", WET-ICE Security Conference, MIT, 2001, <http://www.computer.org/csdl/proceedings/wetice/2001/1269/00/12690150.pdf>.

[FIPS.180-4.2012] National Institute of Standards and Technology, U.S. Department of Commerce, "Secure Hash Standard (SHS)", FIPS 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[H2HIPSEC] Aura, T., Roe, M., and A. Mohammed, "Experiences with Host-to-Host IPsec", 13th International Workshop on Security Protocols, Cambridge, UK, April 2005.

[HMAC] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

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

[IDNA] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

[IKEV1] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998, <http://www.rfc-editor.org/info/rfc2409>.

[IKEV2] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005, <http://www.rfc-editor.org/info/rfc4306>.

[IP] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[IP-COMP] Shacham, A., Monsour, B., Pereira, R., and M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 3173, September 2001, <http://www.rfc-editor.org/info/rfc3173>.

[IPSECARCH-OLD] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998, <http://www.rfc-editor.org/info/rfc2401>.

[IPV6CONFIG] Eronen, P., Laganier, J., and C. Madson, "IPv6 Configuration in Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5739, February 2010, <http://www.rfc-editor.org/info/rfc5739>.

[ISAKMP] Maughan, D., Schneider, M., and M. Schertler, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998, <http://www.rfc-editor.org/info/rfc2408>.

[MAILFORMAT] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008, <http://www.rfc-editor.org/info/rfc5322>.

[MD5] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, <http://www.rfc-editor.org/info/rfc1321>.

[MIPV6] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

[MLDV2] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004, <http://www.rfc-editor.org/info/rfc3810>.

[MOBIKE] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006, <http://www.rfc-editor.org/info/rfc4555>.

[MODES] Dworkin, M., "Recommendation for Block Cipher Modes of Operation", National Institute of Standards and Technology, NIST Special Publication 800-38A 2001 Edition, December 2001.

[NAI] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005, <http://www.rfc-editor.org/info/rfc4282>.

[NATREQ] Aboba, B. and W. Dixon, "IPsec-Network Address Translation (NAT) Compatibility Requirements", RFC 3715, March 2004, <http://www.rfc-editor.org/info/rfc3715>.

[OAKLEY] Orman, H., "The OAKLEY Key Determination Protocol", RFC 2412, November 1998, <http://www.rfc-editor.org/info/rfc2412>.

[PFKEY] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, July 1998, <http://www.rfc-editor.org/info/rfc2367>.

[PHOTURIS] Karn, P. and W. Simpson, "Photuris: Session-Key Management Protocol", RFC 2522, March 1999, <http://www.rfc-editor.org/info/rfc2522>.

[RANDOMNESS] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[REAUTH] Nir, Y., "Repeated Authentication in Internet Key Exchange (IKEv2) Protocol", RFC 4478, April 2006, <http://www.rfc-editor.org/info/rfc4478>.

[REUSE] Menezes, A. and B. Ustaoglu, "On Reusing Ephemeral Keys In Diffie-Hellman Key Agreement Protocols", December 2008, <http://www.cacr.math.uwaterloo.ca/techreports/2008/cacr2008-24.pdf>.

[RFC4945] Korver, B., "The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX", RFC 4945, August 2007, <http://www.rfc-editor.org/info/rfc4945>.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010, <http://www.rfc-editor.org/info/rfc5996>.

[RFC6989] Sheffer, Y. and S. Fluhrer, "Additional Diffie-Hellman Tests for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 6989, July 2013, <http://www.rfc-editor.org/info/rfc6989>.

[ROHCV2] Ertekin, E., Christou, C., Jasani, R., Kivinen, T., and C. Bormann, "IKEv2 Extensions to Support Robust Header Compression over IPsec", RFC 5857, May 2010, <http://www.rfc-editor.org/info/rfc5857>.

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

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

[TRANSPARENCY] Carpenter, B., "Internet Transparency", RFC 2775, February 2000, <http://www.rfc-editor.org/info/rfc2775>.

Приложение A. Отличия от IKEv1

Цели пересмотра IKE перечислены ниже

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 удаленных адресов.
2. Упрощение IKE за счет 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. [EXCHANGEANALYSIS].
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 счет 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 счет 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 счет предоставления ответчику возможности не выполнять существенной обработки до подтверждения инициатором возможности приема сообщений по заявленному им адресу IP.
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, определенных здесь для использования в протоколе IKE. Эти группы создал Richard Schroeppel из Университета штата Аризона. Свойства этих простых чисел описаны в работе [OAKLEY].

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.

Приложение C. Обмены и элементы данных

В этом предложении кратко проиллюстрированы обмены IKEv2 с элементами данных каждого сообщения. Приложение является лишь информативным и при наличии в нем разногласий с текстом основного документа, верным считается основной текст.

Элементы данных Vendor ID (V) могут включаться почти во все сообщения. Ниже они представлены в наиболее логичных местах.

C.1. Обмен IKE_SA_INIT

```
Запрос                --> [N(COOKIE)],
                          SA, KE, Ni,
                          [N(NAT_DETECTION_SOURCE_IP)+,
                           N(NAT_DETECTION_DESTINATION_IP)],
                          [V+][N+]

обычный отклик        <-- SA, KE, Nr,
(без cookie)              [N(NAT_DETECTION_SOURCE_IP),
                           N(NAT_DETECTION_DESTINATION_IP)],
                          [[N(HTTP_CERT_LOOKUP_SUPPORTED)], CERTREQ+],
                          [V+][N+]

отклик с cookie       <-- N(COOKIE), [V+][N+]

нужна другая группа <-- N(INVALID_KE_PAYLOAD),
группа Diffie-Hellman     [V+][N+]
```

C.2. Обмен IKE_AUTH без EAP

```
Запрос                --> IDi, [CERT+],
                          [N(INITIAL_CONTACT)],
                          [[N(HTTP_CERT_LOOKUP_SUPPORTED)], CERTREQ+],
                          [IDr],
                          AUTH,
                          [CP(CFG_REQUEST)],
                          [N(IPCOMP_SUPPORTED)+],
                          [N(USE_TRANSPORT_MODE)],
                          [N(ESP_TFC_PADDING_NOT_SUPPORTED)],
                          [N(NON_FIRST_FRAGMENTS_ALSO)],
                          SA, TSi, TSr,
                          [V+][N+]
```

```
отклик                <-- IDr, [CERT+],
                          AUTH,
                          [CP(CFG_REPLY)],
                          [N(IPCOMP_SUPPORTED)],
                          [N(USE_TRANSPORT_MODE)],
                          [N(ESP_TFC_PADDING_NOT_SUPPORTED)],
                          [N(NON_FIRST_FRAGMENTS_ALSO)],
                          SA, TSi, TSr,
                          [N(ADDITIONAL_TS_POSSIBLE)],
                          [V+][N+]

ошибка при          <--   IDr, [CERT+],
создании Child SA         AUTH, N(error), [V+][N+]
```

C.3. Обмен IKE_AUTH с EAP

```
Первый запрос         --> IDi,
                          [N(INITIAL_CONTACT)],
                          [[N(HTTP_CERT_LOOKUP_SUPPORTED)], CERTREQ+],
                          [IDr],
                          [CP(CFG_REQUEST)],
                          [N(IPCOMP_SUPPORTED)+],
                          [N(USE_TRANSPORT_MODE)],
                          [N(ESP_TFC_PADDING_NOT_SUPPORTED)],
                          [N(NON_FIRST_FRAGMENTS_ALSO)],
                          SA, TSi, TSr,
                          [V+][N+]

первый отклик         <-- IDr, [CERT+], AUTH, EAP, [V+][N+]

                    / --> EAP
повтор 1..N раз     |
                    \ <-- EAP

послед. запрос        --> AUTH

послед. отклик        <-- AUTH,
                          [CP(CFG_REPLY)],
                          [N(IPCOMP_SUPPORTED)],
                          [N(USE_TRANSPORT_MODE)],
                          [N(ESP_TFC_PADDING_NOT_SUPPORTED)],
                          [N(NON_FIRST_FRAGMENTS_ALSO)],
                          SA, TSi, TSr,
                          [N(ADDITIONAL_TS_POSSIBLE)],
                          [V+][N+]
```

C.4. Обмен CREATE_CHILD_SA для создания и смены ключей Child SA

```
Запрос                --> [N(REKEY_SA)],
                          [CP(CFG_REQUEST)],
                          [N(IPCOMP_SUPPORTED)+],
                          [N(USE_TRANSPORT_MODE)],
                          [N(ESP_TFC_PADDING_NOT_SUPPORTED)],
                          [N(NON_FIRST_FRAGMENTS_ALSO)],
                          SA, Ni, [KEi], TSi, TSr
                          [V+][N+]

обычный               <-- [CP(CFG_REPLY)],
отклик                    [N(IPCOMP_SUPPORTED)],
                          [N(USE_TRANSPORT_MODE)],
                          [N(ESP_TFC_PADDING_NOT_SUPPORTED)],
                          [N(NON_FIRST_FRAGMENTS_ALSO)],
                          SA, Nr, [KEr], TSi, TSr,
                          [N(ADDITIONAL_TS_POSSIBLE)]
                          [V+][N+]

ошибка                <-- N(error)

нужна другая группа   <-- N(INVALID_KE_PAYLOAD),
Diffie-Hellman            [V+][N+]
```

C.5. Обмен CREATE_CHILD_SA для смены ключей IKE SA

```
Запрос                --> SA, Ni, Kei [V+][N+]

отклик                <-- SA, Nr, Ker [V+][N+]
```

C.6. Обмен INFORMATIONAL

```
Запрос                --> [N+], [D+], [CP(CFG_REQUEST)]
```

```
отклик                  <-- [N+], [D+], [CP(CFG_REPLY)]
```

Благодарности

Многие члены рабочей группы IPsecME внесли значительный вклад в создание идей и текста настоящего документа, а также в рассмотрение разъяснений, предложенных другими людьми.

Ниже приведена копия раздела благодарностей из первой спецификации IKEv2.

> 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.

Адреса авторов

Charlie Kaufman

Microsoft

1 Microsoft Way

Redmond, WA 98052

United States

EMail: charliekaufman@outlook.com

Paul Hoffman

VPN Consortium

127 Segre Place

Santa Cruz, CA 95060

United States

Phone: 1-831-426-9827

EMail: paul.hoffman@vpnc.org

Yoav Nir

Check Point Software Technologies Ltd.

5 Hasolelim St.

Tel Aviv 6789735

Israel

EMail: ynir.ietf@gmail.com

Pasi Eronen

Independent

EMail: pe@iki.fi

Tero Kivinen

INSIDE Secure

Eerikinkatu 28

HELSINKI FI-00180

Finland

EMail: kivinen@iki.fi

Перевод на русский язык

Николай Малых

nmalykh@gmail.com