Internet Engineering Task Force (IETF)
Request for Comments: 7401
Obsoletes: 5201
Category: Standards Track
ISSN: 2070-1721

R. Moskowitz, Ed.
HTT Consulting
T. Heer
Hirschmann Automation and Control
P. Jokela
Ericsson Research NomadicLab
T. Henderson
University of Washington

April 2015

Host Identity Protocol Version 2 (HIPv2)

Протокол отождествления хостов версии 2 (HIPv2)

Аннотация

Этот документ определяет детали протокола отождествления хостов (Host Identity Protocol или HIP). Протокол HIP позволяет соответствующим хостам безопасно организовывать и поддерживать общее состояние уровня IP, что позволяет разделить роли идентификатора и «локатора» для адресов IP, обеспечивая непрерывность связи при смене адреса IP. Протокол HIP основан на обмене ключами Diffie-Hellman с использованием открытых ключей в качестве идентификаторов из нового пространства имён Host Identity для взаимной аутентификации партнёров. Протокол рассчитан на устойчивость а атакам на отказ в обслуживании (denial-of-service или DoS) и перехвату с участием человека (man-in-the-middle или MitM). При использовании с другим подходящим протоколом защиты, таким как ESP (Encapsulating Security Payload), протокол обеспечивает защиту целостности и может обеспечивать шифрование для протоколов вышележащего уровня (например, TCP или UDP).

Этот документ отменяет RFC 5201 и решает отмеченные IESG проблемы, в частности, вопрос гибкости шифрования. Документ также учитывает опыт реализации RFC 5201.

Статус документа

Документ относится к категории Internet Standards Track.

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 5741.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке http://www.rfc-editor.org/info/rfc7401.

Оглавление

[1] Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2] Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Этот документ задаёт детали протокола отождествления хостов (HIP). Высокоуровневое описание протокола и его архитектуры HIP представлено в отдельном документе [HIP-ARCH]. Вкратце, архитектура HIP предлагает альтернативу двойному использованию адресов IP как «локаторов» (меток маршрутизации) и «идентификаторов» (обозначения конечных точек - хостов). В HIP применяются открытые ключи или пары ключей (открытый и секретный) в качестве идентификаторов хостов, к которым привязаны вышележащие протоколы вместо адресов IP. Благодаря применению открытых ключей (и их представлений) в качестве идентификаторов хостов, хосты могут напрямую аутентифицировать динамическую смены адресов IP, а при желании можно использовать строгую проверку подлинности между хостами на уровне стека TCP/IP.

Этот документ задаёт базовый протокол HIP (базовый обмен), используемый между хостами для создания контекста взаимодействия на уровне IP, называемого ассоциацией HIP, до начала коммуникаций. Документ определяет формат пакетов, а также процедуры обновления и разрыва активных ассоциаций HIP. Архитектура HIP включает другие элементы, описанные в отдельных документах, включая перечисленные ниже.

- «Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)» [RFC7402] описывает применение протокола ESP для защиты целостности и необязательного шифрования.
- «Host Mobility with the Host Identity Protocol» [HIP-HOST-MOB] определяет поддержку мобильности хостов в HIP.
- «Host Identity Protocol (HIP) Domain Name System (DNS) Extension» [HIP-DNS-EXT] определяет расширение DNS для включения данных Host Identity.
- «Host Identity Protocol (HIP) Rendezvous Extension» [HIP-REND-EXT] задаёт использование механизма «встречи» (rendezvous) для связи с мобильными хостами HIP.

С момента разработки базового обмена HIP были достигнуты успехи в криптографии и атаках на криптосистемы, в результате чего потребовалось обеспечить гибкость криптографических протоколов, т. е. возможность переключения с одного криптографического примитива на другой стала частью таких протоколов. Важно поддерживать разумный набор базовых алгоритмов для разных вариантов применения и обеспечить возможность отказа от алгоритмов, для которых были обнаружены уязвимости. Это обновление базового обмена обеспечивает требуемую криптографическую гибкость при борьбе с атаками на понижение, которые становятся возможными в результате гибкости. Кроме того, добавлена поддержка эллиптических кривых с использованием Elliptic Curve DSA (ECDSA) и Elliptic Curve Diffie-Hellman (ECDH).

1.1. Новое пространство имён и идентификаторы

Протокол HIP вводит новое пространство имён Host Identity (отождествление хостов), некоторые аспекты которого рассмотрены в описании архитектуры HIP[HIP-ARCH].

Имеется два основных представления отождествления хоста - полное (Host Identity или HI) и тег отождествления (Host Identity Tag или HIT). HI является открытым ключом и напрямую представляет отождествление хоста. Поскольку имеются разные алгоритмы открытых ключей, использующие разный размер ключа, HI напрямую не подходит в качестве идентификатора пакета или индекса связанных с реализацией структур состояний, требуемых для поддержки HIP. Поэтому применяется хэш от HI или тег отождествления хоста (HIT) в качестве рабочего представления. Тег HIT имеет размер 128 битов и применяется в заголовках HIP и для индексирования соответствующих состояний на конечных хостах. У HIT есть важное свойство защиты - самосертификация (см. 3. Отождествление HI и его структура).

1.2. Базовый обмен HIP (BEX)

Базовый обмен HIP - это двухсторонний криптографический протокол, используемый для организации между хостами коммуникационного контекста. Базовый обмен включает 4 пакета и совместим с SIGMA [KRA03]. Первая сторона обмена называется Инициатором (Initiator), вторая - Ответчиком (Responder). Протокол выполняет обмен ключами Diffie-Hellman (DH) [DIF76] во втором и третьем пакетах, а также аутентифицирует стороны в третьем и четвертом. Применение 4 пакетов повышает устойчивость HIP к DoS-атакам и позволяет Ответчику не устанавливать состояние,

пока не будут проверены адрес IP и криптографическая головоломка (puzzle). Ответчик начинает обмен puzzle по втором пакете, а Инициатор завершает его в третьем до того, как ответчик сохранит какое-либо состояние из обмена.

В обмене может применяться вывод DH для шифрования Host Identity Инициатора в третьем пакете (хотя в работе Aura и др. [AUR05] отмечено, что такая операция может мешать работе промежуточных устройств с проверкой пакетов) или Host Identity можно передать без шифрования. Host Identity Ответчика не защищается. Следует отметить, что теги HIT Инициатора и Ответчика передаются в пакетах открыто, что позволяет перехватчику, имеющему сведения о сторонах взаимодействие, идентифицировать их по тегам HIT. Следовательно, шифрование HI любой из сторон не обеспечивает защиты от таких аткак.

Передача пакетов данных начинается после четвёртого пакета. Третий и четвёртый пакеты HIP в будущем смогут также передавать данные, однако детали этого будут определены позднее.

Существующая ассоциация HIP может быть обновлена с использованием определённого в этом документе механизма, а когда ассоциация становится ненужной, её можно закрыть с помощью определённого в документе механизма.

HIP разработан как протокол сквозной аутентификации и организации ключей для использования с ESP [RFC7402] и другими протоколами сквозной защиты. Базовый протокол не охватывает все детали управления правилами, присутствующие в обмене ключами IKE (Internet Key Exchange) [RFC7296] и позволяющие IKE поддерживать сложные правила на шлюзах. Поэтому HIP не является полнофункциональной заменой IKE.

1.3. Структура документа

В разделе 2 определены некоторые ключевые слова, обозначения и термины, применяемые в документе, в разделе 3 определена структура Host Identity и различных представлений. В разделе 4 приведён обзор протокола базового обмена HIP, а разделы 5 и 6 описывают формат пакетов и правила их обработки. В разделах 7, 8 и 9 обсуждаются правила, безопасность и взаимодействие с IANA.

2. Термины и определения

2.1. Уровни требований

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 RFC 2119 [RFC2119].

2.2. Обозначения

[x]
: Указывает, что значение x является необязательным.

{x}
: Указывает, что значение x является шифрованным.

X(y)
: Указывает, что y является параметром X.

<x>i
: Указывает, что имеется i элементов x.

-->
: Связь от Инициатора к Ответчику (запросы).

<--
: Связь от Ответчика к Инициатору (отклики).

|
: Объединение (конкатенация).

Ltrunc (H(x), #K)
: Указывает #K младших битов результата хэш-функции H от входных данных x.

2.3. Определения

HIP base exchange (BEX) - базовый обмен HIP
: Согласование для организации новой ассоциации HIP.

Host Identity (HI) - отождествление хоста
: Открытый ключ алгоритма подписи, представляющий отождествление хоста. В HIP хост подтверждает своё отождествление, создавая подпись с секретным ключом, относящимся к его HI (см. 3. Отождествление HI и его структура).

Host Identity Tag (HIT) - тег отождествления хоста
: Сокращение HI в формате IPv6, создаваемое путём хэширования HI (см. 3.1. Тег отождествления хоста (HIT)).

HIT Suite - комплект HIT
: HIT Suite группирует все криптоалгоритмы, которые нужны для генерации и применения HI и его тега HIT. В частности, это 1) алгоритм подписи с открытым ключом, 2) хэш-функция и 3) отсечка (см. Приложение E. HIT Suite и генерация HIT).

HIP association - ассоциация HIP
: Общее состояние двух партнёров после завершения BEX.

HIP packet - пакет HIP
: Пакет управления с заголовком HIP, относящийся к организации, поддержке или прерыванию ассоциации HIP.

Initiator - Инициатор
: Хост, инициирующий BEX. Эта роль обычно теряется по завершении BEX.

Responder - Ответчик
: Хост, отвечающий Инициатору BEX. Эта роль обычно теряется по завершении BEX.

Responder's HIT hash algorithm (RHASH) - алгоритм хэширования HIT ответчика
: Алгоритм хэширования, применяемый в документе для разных расчётов хэш-значений, а также для генерации тега HIT Ответчика. RHASH является хэш-функцией, определяемой HIT Suite тега HIT Ответчика (см. 5.2.10. HIT_SUITE_LIST).

Length of the Responder's HIT hash algorithm (RHASH_len) - размер вывода алгорита RHASH
Естественный размер вывода RHASH в битах.

Signed data - подписанные данные
Подписанные данные защищаются цифровой подписью, созданной отправителем с применением секретного ключа его HI.

KDF - функция вывода ключей
Функция вывода ключей (Key Derivation Function или KDF) служит для создания симметричных ключей в обмене DH.

KEYMAT - ключевой материал
Ключевой материал, выведенный из обмена ключами DH с использованием функции KDF. Симметричные ключи для шифрования и защиты целостности пакетов HIP и шифрованных пакетов данных пользователя выводятся из этого ключевого материала.

3. Отождествление HI и его структура

В этом разделе рассматриваются свойства Host Identity и Host Identity Tag, а также определяется их точный формат. В протоколе HIP открытый ключ асимметричной пары ключей служит отождествлением хоста HI. Соответственно, сам хост определяется как объект, содержащий секретный ключ пары. Более подробное описание различий между отождествлением и соответствующим идентификатором приведено в документе по архитектуре HIP [HIP-ARCH].

Реализации HIP должны поддерживать алгоритм открытых ключей RSA (Rivest Shamir Adleman) [RSA] и алгоритм ECDSA (Elliptic Curve Digital Signature Algorithm) для генерации HI, как указано в параграфе 5.2.9. Могут также поддерживаться дополнительные алгоритмы.

Хешированное кодирование HI - тег HIT - применяется в протоколах для представления Host Identity. HIT имеет размер 128 битов и три важных свойства: i) размер совпадает с размером адреса IPv6, что позволяет применять его в адресных полях API и протоколов с фиксированным размером, ii) тег сам сертифицирует себя (т. е. по данному HIT сложно расчетным путём найти соответствующий ключ Host Identity) и iii) вероятность совпадения HIT у двух хостов (конфликт) очень мала, поэтому злоумышленнику не удастся найти конфликт с используемым тегом HIT. Более подробное описание свойств безопасности HIT приведено в [HIP-ARCH].

Структура HIT определена в [RFC7343]. HIT - это наложенный маршрутизируемый криптографический хэш-идентификатор (Overlay Routable Cryptographic Hash Identifier или ORCHID), состоящий из 3 частей. Назначаемый IANA префикс служит для того, чтобы отличать теги от адресов IPv6. 4-битовый код указывает алгоритм, применяемый для генерации HI и хэшированного представления HI. Третьей частью является 96-битовое хэш-представление Host Identity. Кодирование алгоритма генерации ORCHID и точный алгоритм генерации хэш-представления заданы в Приложении E и [RFC7343].

Передача HI и HIT в заголовках пакетов пользовательских данных повышает издержки для этих пакетов, поэтому передача указанных значений не предполагается в каждом пакете и применяются иные методы сопоставления пакетов с HI. В некоторых случаях можно применять HIP без дополнительных заголовков в пользвательских пакетах данных. Например, при использовании ESP для защиты трафика данных индекс параметров защиты (Security Parameter Index или SPI) в заголовке ESP может служить для сопоставления защифрованного пакета данных с ассоциацией HIP.

3.1. Тег отождествления хоста (HIT)

128-битовый тег HIT - это хэшированное значение HI. Использование хэшированного представления идентификатора обеспечивает два преимущества по сравнению открытым ключом отождествления хоста, имеющим переменный размер. Во-первых, фиксированный размер HIT позволяет сохранить контроль за размером пакетов и упрощает кодирование протоколов. Во-вторых, он обеспечивает согласованный формат протокола, не зависящий от применяемой технологии отождествления.

RFC 7343 [RFC7343] задаёт 128-битовые идентификаторы на основе хэш-значений, называемые ORCHID. Префикс для этих идентификаторов, выделенный из блока адресов IPv6, также определён в [RFC7343]. Тег HIT является одним из типов ORCHID.

Этот документ расширяет исходную, экспериментальную спецификацию HIP [RFC5201] мерами обеспечения криптографической гибкости. Одна из таких мер разрешает использовать разные хэш-функции для генерации тегов HIT. HIT Suite группирует наборы алгоритмов, которые нужны для создания и применения конкретного HIT, и указываются эти группы с помощью HIT Suite ID, передаваемых в поле алгоритма OGA (ORCHID Generation Algorithm) идентификатора ORCHID. С помощью HIT Suite ID в поле OGA ID хост может определить по тегу HIT другого хоста, поддерживает ли тот необходимые алгоритмы хэширования и подписи для создания ассоциации HIP с этим хостом.

3.2. Создание HIT из HI

Теги HIT **должны** создаваться в соответствии с методом генерации ORCHID, описанным в [RFC7343], с идентификатором контекста 0xF0EF F02F BFF4 3D0F E793 0C3C 6E61 74EA (случайное значение, созданное редактором данной спецификации) и входных данных, представляющих поле Host Identity (см. параграф 5.2.9) в данных пакета HIP. Хэш-функцию, алгоритм подписи и алгоритм генерации HIT из HI определяет набор HIT Suite (см. параграф 5.2.10), который указывается 4 битами поля OGA ID в идентификаторе ORCHID. В настоящее время для создания HIT определены усеченные алгоритмы SHA-1, SHA-384 и SHA-256 [FIPS.180-4.2012].

Для отождествлений, являющихся открытыми ключами RSA, DSA (Digital Signature Algorithm) [FIPS.186-4.2013] или ECDSA (Elliptic Curve DSA), вводом ORCHID служит кодирование открытого ключа, указанное в поле Host Identity параметра HOST_ID (см. параграф 5.2.9). Этот документ определяет 4 алгоритма: RSA, DSA, ECDSA, ECDSA_LOW. Профиль ECDSA_LOW предназначен для устройств с малой вычислительной мощностью, поэтому применим однин из указанных ниже вариантов.

Открытый ключ RSA кодируется в соответствии с разделом 2 в [RFC3110], принимая конкатенацию полей размера экспоненты (e_len), показателя (e) и модуля (n). Размер n_len модуля n может быть определён из общего размера HI и предшествующих полейHI, включая показатель степени (e). Таким образом, входные данные для генерации HIT имеют такой же размер, как HI. Поля **должны** кодироваться в сетевом порядке байтов, как указано в [RFC3110].

Открытый ключ DSA кодируется в соответствии с разделом 2 в [RFC2536], принимая конкатенацию полей T, Q, P, G, Y. Таким образом, размер хэшируемых данных составляет 1 + 20 + 3 * 64 + 3 * 8 * T октетов, где T - параметр размера, определённый в [RFC2536]. Этот параметр влияет на размеры полей, поэтому для него **должно** выбираться минимальное значение, позволяющее поместить P, G и Y. Поля **должны** кодироваться в сетевом порядке байтов, как указано в [RFC2536].

Открытые ключи ECDSA кодируются в соответствии с параграфом 4.2 и разделом 6 в [RFC6090].

В Приложении B представлен псевдокод, иллюстрирующий кодирование открытых ключей.

4. Обзор протокола

В этом разделе приведено упрощенно описание работы протокола HIP без рассмотрения формата пакетов и этапов обработки, которые приведены в разделах 5 и 6, являющихся нормативными.

Для протокола HIP агентство IANA выделило номер 139.

Заголовок данные HIP (параграф 5.1) может передаваться в каждой дейтаграмме IP. Однако эти заголовки достаточно велики (40 байтов), поэтому желательно «сжимать» заголовки HIP, передавая их лишь в пакетах управления при организации или смене состояния ассоциации HIP. Фактический метод «сжатия» и сопоставления пакетов с имеющимися ассоциациями (при их наличии) определяется отдельными документами, описывающими форматы и методы транспортировки. Все реализации HIP **должны** поддерживать транспортный формат ESP для HIP [RFC7402].

4.1. Создание HIP-ассоциации

Система, начинающая базовый обмен HIP считается Инициатором (Initiator), а её партнёр - Ответчиком (Responder). Это различие обычно утрачивается по завершении базового обмена и любая их сторон может стать Инициатором в будущих коммуникациях.

Базовый обмен HIP служит для организации состояния между Инициатором и Ответчиком. Первый пакет (I1) начинает обмен, а 3 оставшихся (R1, I2, R2) реализуют аутентифицированный обмен ключами Diffie-Hellman (DH) [DIF76] для генерации сеансовых ключей. В двух первых пакетах хосты согласуют набор криптографических идентификаторов и алгоритмы, которые применяются после обмена. В процессе обмена ключами DH создаётся фрагмент ключевого материала, из которого выводятся ключи ассоциации HIP с помощью функции вывода ключей (Key Derivation Function или KDF). Если нужны другие криптографические ключи, например, для использования с ESP, предполагается их вывод из того же ключевого материала с использованием KDF.

Инициатор сначала передаёт Ответчику запускающий пакет I1, который содержит его тег HIT и может включать HIT Ответчика, если этот тег известен. Кроме того, пакет I1 инициализирует согласование группы DH, используемой для создания ключевого материала, поэтому пакет I1 включает список идентификаторов групп DH, поддерживаемых Инициатором. Отметим, что в некоторых случаях возможна замена этого пакета иным триггером и в таких случаях протокол начинает с передачи Ответчиком пакета R1, при этом должен указываться механизм передачи списка поддерживаемых Инициатором групп DH (например, использование принятой по умолчанию группы).

Второй пакет (R1) фактически начинает аутентифицированный обмен DH. Этот пакет содержит «головоломку» (puzzle), являющуюся криптографической задачей которую Инициатор должен решить для продолжения обмена. Уровень сложности этой задачи можно настраивать в зависимости от степени доверия к Инициатору, текущей нагрузки и других факторов. В дополнение к этому R1 содержит параметр DH для Ответчика и список поддерживаемых Ответчиком криптографических алгоритмов. На основе этого Инициатор может продолжить, прервать или перезапустить базовый обмен с соответствующим набором криптографических алгоритмов. Пакет R1 также включает подпись для выбранных частей сообщения. Некоторые поля не включаются в подпись для поддержки создаваемых заранее пакетов R1.

В пакете I2 Инициатор **должен** указать решение головоломки и при отсутствии верного решения пакет I2 будет отброшен. I2 включает также параметр DH с информацией, требуемой Ответчику. Инициатор подписывает пакет I2.

Пакет R2 подтверждает получение I2 и завершает базовый обмен. Ответчик подписывает этот пакет.

Базовый обмен показан на рисунке 1, где key указывает открытый ключ Host Identity, a sig - подпись с использованием этого ключа. Некоторые параметр пакета на рисунке не показаны.

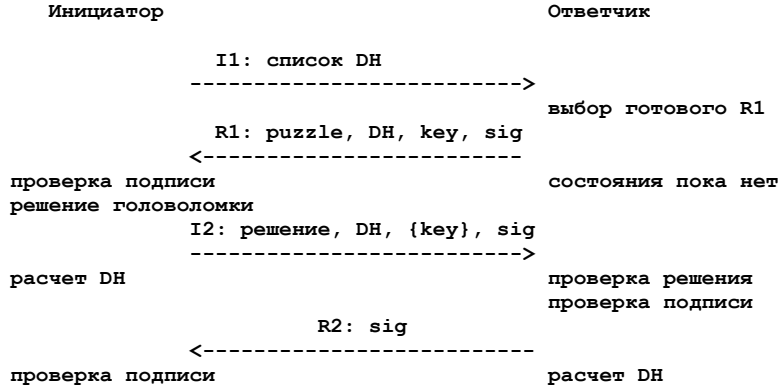


Рисунок 1. Базовый обмен HIP (BEX).

4.1.1. Механизм HIP Puzzle

Механизм HIP Puzzle предназначен для защиты Ответчика от DoS и позволяет Ответчику задержать создание состояния до получения пакета I2. Кроме того, головоломка позволяет Ответчику использовать достаточно простые расчеты для проверки «искренности» Инициатора, требующие от того нескольких тактов CPU для решения задачи.

Головоломка позволяет Ответчику отложить создание состояния ассоциации до получения действительного пакета I2. Пакет I2 без верного решения задачи может быть отвергнут сразу после проверки решения Ответчиком. Решение

можно проверить расчетом лишь одной хэш-функции, а непригодные решения могут быть отвергнуты до создания состояния и требующих ресурсов CPU проверки подписи и создания ключей DH. Меняя сложность головоломки, Ответчик может предотвратить DoS-атаки, нацеленные на рсход ресурсов CPU или памяти.

Ответчик может оставаться без состояния и отбрасывать большинство обманных пакетов I2, поскольку для решения задачи нужен HIT-тег Инициатора. Идея состоит в том, что у Ответчика имеется несколько (возможно переменное число) готовых пакетов R1, из которых он выбирает подходящий на основе сведений из пакета I1. Когда Ответчик получает пакет I2, он может проверить решение задачи с использованием HIT Инициатора. Это делает бессмысленным для атакующего сначала передать один пакет I1/R1, а затем генерировать множество фиктивных пакетов I2, как будто с разными HIT. Однако метод не защищает Ответчика от атак с использованием фиксированного значения HIT, против которых может применяться частичное создание локального состояния с запоминанием фактов отказа при проверке решения (один из вариантов описан в Приложении A). Реализациям Ответчика **следует** включать достаточный объем случайности в значения puzzle, чтобы предотвратить алгоритмические атаки [CRO03].

Ответчик может устанавливать сложность головоломки для Инициатора на основе уровня доверия к нему. Поскольку головоломка не учитывается при создании подписи, Ответчик может использовать созданные заранее пакеты R1 и передавать их в качестве головоломки для Инициатора. Ответчику **следует** применять эвристические методы обнаружения DoS-атак и устанавливать сложность головоломки (#K), как описано ниже.

4.1.2. Обмен Puzzle

Ответчик запускает обмен Puzzle, получив пакет I1. Ответчик предоставляет случайное значение #I и требует от Инициатора найти случайное значение #J. Для выбора нужного #J Инициатор должен создать конкатенацию #I, тегов HIT обеих сторон и #J, затем рассчитать хэш этой конкатенации, используя алгоритм RHASH. Младшие #K результата должны быть 0. Значение #K определяет сложность головоломки.

Для генерации верного #J Инициатор создаёт множество значений #J, пока не будет получена хеш-цель из нулей. Инициатору **следует** прерывать попытки только по истечении срока действия головоломки, заданного полем Lifetime в параметре PUZZLE (см. параграф 5.2.4). Ответчику требуется восстановить конкатенацию #I, тегов HIT и представленного #J, а затем рассчитать хэш для проверки корректности решения задачи Инициатором.

Для предотвращения атак с предварительным расчетом Ответчик **должен** выбирать значение #I так, чтобы Инициатор не мог угадать его. Кроме того, конструкция **должна** позволять Ответчику убедиться в том, что значение #I выбрано им, а не Инициатором. Пример реализации этого приведён в Приложении A.

Используя поле Opaque в PUZZLE (см. параграф 5.2.4) параметра ECHO_REQUEST_SIGNED (см. параграф 5.2.20) или ECHO_REQUEST_UNSIGNED (см. параграф 5.2.21), Ответчик может включить в R1 те или иные данные, которые Инициатор **должен** без изменений скопировать в соответствующий пакет I2. Ответчик может использовать необрабатываемые (opaque) данные для передачи части сведений локального состояния Инициатору и обратно, например, для проверки того, что I2 является откликом на переданный ранее R1. Ответчик может создавать эти данные разными способами, например, применяя шифрование или хэширование с неким секретом, переданное значение #I и возможно другие связанные данные. Используя тот же секрет, полученное значение #I (из пакета I2) и другие связанные данные (при наличии), Ответчик может убедиться, что это он передал #I Инициатору. Ответчик **должен** периодически менять применяемый секрет.

Ответчику **рекомендуется** генерировать новые секреты для головоломки и новые R1 каждые несколько минут. Кроме того, ему **рекомендуется** обеспечивать возможность проверки правильности решения головоломки в течение не меньше Lifetime секунд после смены секрета головоломки. Это гарантирует действительность головоломки в интервале от Lifetime до 2 * Lifetime секунд. Это не позволит атакующему воспользоваться старой головоломкой и предотвратит проблемы в сучаях, когда срок действия значение Lifetime сравнимо с задержкой в сети и временем решения головоломки.

Значение головоломки #I и решение #J служат входными данными для создания ключевого материала из обмена DH (см. параграф 6.5). Поэтому для того, чтобы ключевой материал различался, Ответчику не следует дважды использовать одно значение #I с теми же ключами DH для одного Инициатора. Такую уникальность можно обеспечить, например, с использованием счётчика в качестве дополнения входных данных при создании #I. Этот счётчик можно увеличивать для каждого обработанного пакета I1. Состояние счётчика можно передавать в поле Opaque в PUZZLE (см. параграф 5.2.4) параметра ECHO_REQUEST_SIGNED (см. параграф 5.2.20) или ECHO_REQUEST_UNSIGNED (см. параграф 5.2.21) без необходимости организации состояния.

Примечание. Разработчики протокола явно рассмотрели вопрос включения в R1 временной метки для защиты Инициатора от атак с повторным использованием (replay). Принято решение не включать метку для предотвращения проблем с глобальной синхронизацией часов.

Примечание. Разработчики протокола явно рассмотрели вопрос использования привязки функции для головоломки к памяти, а не к CPU. Принято решение не использовать привязанные к памяти функции.

4.1.3. Аутентифицированный обмен Diffie-Hellman с согласованием DH Group

Пакеты R1, I2, R2 реализуют стандартный аутентифицированный обмен DH. Ответчик передаёт один из своих открытых ключей DH и свой открытый ключ аутентификации (т. е. Host Identity) в пакете R1. Подпись в пакете R1 позволяет Инициатору убедиться в том, что пакет R1 создан Ответчиком. Однако, поскольку R1 создаётся заранее и не охватывает связанную с ассоциацией информацию из пакета I1, это не обеспечивает защиты от replay-атак.

Перед фактическим аутентифицированным обменом DH Инициатор указывает своё предпочтение по выбору групп DH в пакете I1 как отсортированный список идентификаторов групп DH. Пакет I1 не подписывается и , следовательно, список передаётся без аутентификации для предотвращения ресурсоемких расчётов при обработке пакета I1 на стороне Ответчика. На основе предпочтений Инициатора Ответчик указывает в пакете R1 наиболее подходящее открытое значение DH, а также присоединяет список своих предпочтений к пакету R1, чтобы передать Инициатору основания для выбора группы DH. Этот список передаётся в подписанной части пакета R1. Если выбор группы DH в пакете R1 не соответствует предпочтениям Инициатора и Ответчика, Инициатор может счесть, что список DH Group ID в пакете I1 был изменён (см. ниже).

Если Ответчик не поддерживает ни один из DH Group ID в пакете I1, он выбирает наиболее подходящую для себя группу DH, назависимо от предпочтений Инициатора, затем передаёт Инициатору пакет R1 с этой группой DH и списком поддерживаемых DH Group ID.

Инициатор получает в пакете R1 открытые значение DH и список групп DH Group ID, поддерживаемых Ответчиком. Этот список включается в подпись пакета R1 для предотвращения подмены. Инициатор сравнивает Group ID открытого значения DH в пакете R1 со списком поддерживаемых DH Group ID в пакетах R1 и своими предпочтениями, указанными в списке поддерживаемых DH Group ID. Инициатор продолжает обмен BEX лишь в том случае, когда Group ID открытого значения DH Ответчика является идентификатором наиболее предпочтительной группы, поддерживаемой Инициатором и Ответчиком. Иное говорит о воздейтсиии на ассоциацию атаки с понижением и Инициатор должен перезапустить базовый обмен новым пакетом I1 или прервать этот обмен. Если Ответчик выбрал группу DH, не поддерживаемую Инициатором, тот **может** прервать согласование или передать новый пакет I1 с другим списком поддерживаемых групп DH. Однако Инициатор **должен** проверить подпись пакета R1 до перезапуска или прерывания согласования. При некорректной подписи Инициатор **должен** просто игнорировать пакет R1.

Если предпочтения в части DH Group ID совпадают, Инициатор рассчитывает сеансовый ключ DH (Kij) и создаёт ассоциацию HIP используя материал из сеансового ключа (см. параграф 6.5). Ассоциация может также использоваться Инициатором для шифрования своего открытого ключа аутентификации, т. е. Host Identity. Результирующий пакет I2 содержит ключ DH Инициатора и его (возможно зашированный) открытый ключ аутентификации. Подпись сообщения I2 охватывает все параметры подписываемых диапазонов (см. параграф 5.2) без исключений, как для пакета R1.

Ответчик извлекает открытый ключ DH Инициатора из пакета I2, рассчитывает сеансовый ключ DH, создаёт соответствующую ассоциацию HIP и расшифровывает открытый ключ аутентификации Инициатора. После этого он может проверить подпись с использованием ключа аутентификации.

Финальное сообщение R2 завершает обмен BEX и защищает Инициатора от replay-атак, поскольку Ответчик использует общий ключ из обмена DH для создания хэшированного кода аутентификации сообщения (Hashed Message Authentication Code или HMAC), а также использует секретный ключ своего Host Identity для подписания пакета.

4.1.4. Защита HIP от повторного использования пакетов (Replay)

HIP включает описанные здесь механизмы для защиты от враждебного повторного использования пакетов (replay). Ответчики защищаются от повторного использования пакетов I1 благодаря ответам на I1 без сохранения состояния подписанными пакетами R1. Инициаторы защищены от повторов R1 с помощью монотонно возрастающих «счетчиков генерации R1» включаемых в пакеты R1. Ответчики защищены от воспроизведения обманных пакетов I2 механизмом головоломок (puzzle, см. параграф 4.1.1) и необязательными данными Opaque. Хосты защищены от повторного использования пакетов R2 и UPDATE применением менее дорогостоящей проверки HMAC до проверки подписи HIP.

Счетчик генерации R1 представляет собой монотонно возрастающее 64-битовое число, для которого может быть установлено любое начальное значение. Счётчик **может** работать на уровне всей системы, однако **следует** поддерживать отдельный счётчик для каждого отождествления Host Identity, если их несколько. Значение этого счётчика **следует** сохранять при перезагрузке системы и вызовах базового обмена HIP. Счётчик указывает текущее поколение головоломок. Реализации **должны** воспринимать головоломки текущего поколения и **могут** воспринимать их из предыдущих поколений. Локальный счётчик системы **должен** инкрементироваться не реже чем прекращается действие каждого прежнего R1. Локальному счетчику **следует** быть неснижаемым, поскольку в противном случае хост предоставит своим партнёрам возможность повторно использовать созданные ранее R1 с большими номерами.

Хост может получит несколько пакетов R1 в результате отправки нескольких I1 (см. параграф 6.6.1) или повторного использования старых R1. При передаче нескольких пакетов I1 одному хосту Инициатору **следует** выждать некоторое время (разумное значение - удвоенный ожидаемый интервал RTT) после получения первого R1, чтобы принять другие R1, а также **следует** отвечать на пакет R1 с наибольшим значением счётчика генерации R1. Если Инициатор обрабатывает R1 или уже передал пакет I2 (ждёт получения R2) и получает другой пакет R1 с большим значением счётчика, он **может** перезапустить обработку R1 для нового пакета R1, как будто он прибыл первым.

Счётчик генерации R1 может достигнуть максимума или быть сброшен. Для Инициатора важна устойчивость к потере состояния или сбросу счётчика генерации R1 у партнёра. При выборе между несколькими R1 Инициатору рекомендуется предпочитать пакет R1 соответствующий текущему счетчику, но если это не работает, Инициатор может воспользоваться другими R1, начиная с пакета с наибольшим значением счетчика.

4.1.5. Отклонение базового обмена HIP

Поддерживающий HIP хост может не воспринимать базовый обмен HIP. Если политика хоста разрешает ему выступать лишь Инициатором и позволяет организовать ассоциацию HIP с исходным Инициатором, хосту следует самому начать базовый обмен HIP. Хост **может** выбрать такую политику, поскольку при обмене защищён лишь идентификатор HI инициатора. Следует отметить, что такое поведение может создавть риск состояния, когда каждый из хостов готов быть лишь инициатором и оба отвергают базовый обмен.

Если политика хоста не позволяет ему вступать в обмен HIP с Инициатором, хосту следует передать сообщение ICMP Destination Unreachable, Administratively Prohibited. Более сложный пакет HIP здесь не применяется, поскольку он открыват больше возможных DoS-атак, чем простое сообщение ICMP. Сообщение HIP NOTIFY не применяется, поскольку между хостами ещё нет ассоциации HIP.

4.1.6. Прерывание базового обмена HIP

Два хоста HIP могут столкнуться с ситуацией, когда они не могут завершить базовый обмен HIP из-за недостаточной поддержки криптографических алгоритмов, в частности HIT Suite и групп DH. После приёма пакета R1 Инициатор может определить, поддерживает ли Ответчик требуемые криптографические операции для создания ассоциации HIP. Инициатор может прервать обмен BEX после получения пакета R1, указывающего неподдерживаемый набор алгоритмов. Конкретные условия описаны ниже.

Пакет R1 содержит подписанный список идентификаторов HIT Suite, поддерживаемых Ответчиком, поэтому Инициатор может определить, поддерживает ли Ответчик тег HIT источника. Если HIT Suite ID из HIT Инициатора не содержится в

списке HIT Suite пакета R1, Инициатор **может** прервать согласования, а также **может** перезапустить его, передав новый пакет I1 с тегом HIT источника, поддерживаемым Ответчиком.

В процессе согласования Инициатор и Ответчик выбирают одну группу DH. Ответчик выбирает группу DH и её открытое значение DH в R1 на основе списка DH Group ID в пакете I1. Если Ответчик не поддерживает ни одной группы DH, указанной Инициатором, он выбирает DH произвольно и отвечает пакетом R1 со списком поддерживаемых DH Group ID. В этом случае Инициатор получает пакет R1 с открытым значением DH для незапрошенной группы DH и список групп DH Ответчика в подписанной части пакета R1. В этот момент Инициатор **может** прервать согласования, а также **может** запустить его заново, передав новый пакет I1 с набором DH Group ID, поддерживаемых Ответчиком.

4.1.7. Защита HIP от понижения

В атаках на понижение (downgrade) злоумышленник пытается незаметно манипулировать пакетами Инициатора и/или Ответчика с целью повлиять на результат криптографического согласования в процессе BEX в свою пользу. В результате жертвы могут выбрать более слабые криптографические алгоритмы, нежели они могли выбрать без вмешательства злоумышленника. Атаки с понижением могут быть успешны лишь в том случае, когда они остаются незамеченными и жертвы считают коммуникационный канал защищенным.

В HIP почти все пакеты параметров, связанные с криптографическим согласованием, имеют подпись. Этими параметрами нельзя манипулировать напрямую с атаках с понижением без аннулирования подписей. Однако подписанные пакеты могут подвергаться replay-атакам, где злоумышленник может воспользоваться старым пакетом BEX с устаревшим и слабым выбором криптоалгоритмов и передать такой пакет вместо свежего с набором более сильных алгоритмов. Такие атаки возможны на пакеты R1 и I2. Однако вопроизведенные пакеты R1 и I2 невозможно применить для успешного обмена HIP BEX, поскольку эти пакеты включают открытые значения DH Инициатора и Ответчика. Старые значения DH из воспроизводимых пакетов ведёт к недействительному ключевому материалу и несоответствию общих секретов, поскольку атакующий не может вывести действительный ключевой материал из открытых ключей DH в R1 и не может создать действительный код HMAC и подпись для воспроизводимого I2.

В отличие от первой версии HIP [RFC5201], версия 2, определённая в этом документе, начинает согласование групп DH уже в первом пакете BEX (I1). Пакеты I1 намеренно не защищены подписью для предотвращения загружающих CPU криптографических операций обработки лавины пакетов I1, направленных Ответчику. Поэтому список DH Group ID в пакете I1 можно подменить или изменить. Для предотвращения незаметных манипуляция с пакетами I1 Ответчик выбирает группу DH детерминированно и включает свой список DH Group ID в подписанную часть пакета R1. Инициатор может заметить попытку понижения, сравнивая список DH Group ID в пакете R1 со своими предпочтениями в пакете I1. Если выбор групп DH в пакете R1 не соответствует лучшему варианту из двух списков (DH ID Ответчика с высшим приоритетом имеется в списке DH от Инициатора), Инициатор может сделать заключение о подмене списка в пакете I1. В этом случае Инициатор может прервать обмен BEX или запустить его заново. Как отмечено выше, обнаружения downgrade-атаки достаточно для её предотвращения.

4.1.8. Режим HIP Opportunistic

Можно инициировать обмен HIP BEX, даже когда HI (и HIT) Ответчика неизвестен. В этом случае в начальном пакете I1 для HIT адресата указывается значение 0. Такой тип соединения называется режимом Opportunistic. Ответчик может иметь несколько HIT в результате поддержки нескольких HIT Suite. Поскольку HIT Suite Ответчика в режиме Opportunistic не определяется HIT получателя в пакете I1, Ответчик волен выбрать HIT любого из своих HIT Suite. Полный набор поддерживаемых Инициатором HIT Suite неизвестен Ответчику, поэтому ему **следует** выбирать HIT из того же HIT Suite, что и HIT Инициатора (указывается информацией HIT Suite в поле OGA ID тега HIT Инициатора), поскольку этот набор очевидно поддерживается Инициатором. Если Ответчик выбирает друго тег HIT, который не поддерживается Инициатором, Инициатор **может** перезапустить BEX пакетом I1 с HIT источника, содержащимся в списке HIT Suite Ответчика из пакета R1.

Отметим, что Инициатор не может проверить подпись пакета R1, если он не поддерживает HIT Suite Ответчика. Поэтому Инициатор **должен** считать пакеты R1 с неподдерживаемыми HIT Ответчика потенциально обманными и ему **недопустимо** использовать из непроверенных R1 какие-либо параметры за исключением HIT_SUITE_LIST. Кроме того, Инициатор, использующий HIT_SUITE_LIST from из непроверенного пакета R1 для определения возможного HIT источника **должен** убедиться, что HIT_SUITE_LIST из первого непроверенного пакета R1 совпадает с HIT_SUITE_LIST во тором пакете R1, для которого Инициатор поддерживает алгоритм подписи. Инициатор **должен** перезапустить обмен BEX новым пакетом I1, для которого алгоритм был указан в проверяемом R1, если два списка различаются. Эта процедура должна смягчить атаки на понижение.

С режимом Opportunistic связаны проблемы как безопасности, так и API, рассматриваемые ниже.

С учётом того, что Инициатор не знает HI Ответчика, должны быть подходящие вызовы API, позволяющие Инициатору напрямую или опосредованно запросить у базовой системы инициирование базового обмена HIP на основе «локаторов» (адресов). HI ответчика будет предварительно представлен в пакете R1 и аутентифицирован после получения и проверки пакета R2. Следовательно, HIT Ответчика можно передать приложению через новые механизмы API. Однако приложение с совместимым с прежними версиями API видит лишь локаторы, использованные при начальном контакте. В зависимости от желаемой семантики API, возникает ряд приведённых ниже проблем.

- Фактические локаторы могут измениться, если применяется сообщение UPDATE, даже при сохранении с точки зрения API ассоциации между двумя конкретными локаторами. Однако обновление локатора остаётся безопасным и ассоциация сохраняется между теми же узлами.

- Разные ассоциации между одной парой локаторо могут приводить к соединениям между разными узлами, если реализация больше не помнит идентификаторов, которые были использованы в более ранней ассоциации. Это может произойти при легитимной изменении локатора партнёра или при захвате атакующим локатора партнёра. Поэтому при использовании режима Opportunistic реализации HIP **недопустимо** предполагать, что HI в сообщении R1 совпадает с HI, ранее показанным для этого адреса.

  Если у реализации HIP и приложения нет единого представления о составе ассоциации, это может происходить даже в рамках одной ассоциации. Например, реализация может не знать, когда следует сбросить состояние HIP для приложения на основе UDP.

Кроме того, здесь применим ряд соображений безопасности. Механизм счётчиков генерации будет менее эффективен для защиты от повторного использования пакетов R1 с учётом того, что Ответчик может выбрать воспроизведение произвольного HI, а не только полученного в пакете. Более важно, что обмен Opportunistic уязвим к MitM-атакам, поскольку у Инициатора нет сведений об открытом ключе партнёра. Чтобы оценить влияние этой уязвимости, она сравнивается с уязвимости современных коммуникаций без HIP.

Атакующий на пути между партнёрами может включиться в процесс (MitM), представляя свой идентификатор Инициатору, а затем создавая другую ассоциацию HIP в направлении Ответчика. Это возможно, если Инициатор использует режим Opportunistic, а Ответчик настроен на восприятие соединений от любого узла с поддержкой HIP. Атакующий вне пути не сможет организовать такую атаку, поскольку он не способен отвечать на сообщения базового обмена.

Эти свойства безопасности характерны для коммуникаций в современной сети Internet. Клиент, связывающийся с сервером без использования сквозной защиты, может обнаружить, что он взаимодействует с сервером через MitM в предположении готовности сервера взаимодействовать с кем угодно. При использовании сквозной защиты худшим, что может случиться как при использовании Opportunistic HIP так и в режиме без HIP (обычный IP) является отказ в обслуживании, когда элемент пути нарушает взаимодействие но не способен выполнить MitM-атаку.

Однако по завершении обмена Opportunistic протокол HIP обеспечивает защиту целостности и конфиденциальности и позволяет безопасно менять локаторы (адреса) конечных точек. В результате режим Opportunistic в HIP предлагает модель защиты «лучше, чем ничего» (better than nothing). Исходно базовый обмен, аутентифицированный в режиме Opportunistic, полагается на веру (leap of faith) и подвержен MitM-атакам, но последующие дейтаграммы, относящиеся к той же ассоциации HIP не могут быть скомпрометированы новой MitM-атакой. Кроме того, при уходе MitM с пути активной ассоциации атака будет обнаружена постфактум. Таким образом, можно считать, что режим Opportunistic в HIP не менее защищён, чем обычные коммуникации IP без защиты.

4.2. Обновление ассоциации HIP

Ассоциацию HIP между парой хостов может потребоваться обновить с течением времени. Примеры включают необходимость смены ключей защищённых связей, новые защищённые связи, смену адресов IP. Для этих о похожих целей применяются пакеты UPDATE. Этот документ задаёт лишь формат и базовые правила обработки UPDATE с обязательными параметрами, а фактическое использование определяется в отдельных спецификациях.

HIP включает пакеты общего назначения UPDATE, которые могут передавать несколько параметров HIP для обновления состояния HIP между парой хостов. В сообщения UPDATE включаются монотонно возрастающие порядковые номера и партнёр явно подтверждает эти сообщения. Потеря UPDATE или подтверждения может быть восстановлена путём повтора передачи. При некоторых обстоятельствах может применяться несколько сообщений UPDATE.

Пакеты UPDATE защищены с помощью параметров HIP_MAC и HIP_SIGNATURE, поскольку обработка подписей UPDATE открывает возможность организации DoS-атак на промежуточные системы.

Пакеты UPDATE подтверждаются явно с помощью параметра, возвращающего полученный от партнера порядковый номер. Один пакет UPDATE может содержать как порядковый номер, так и 1 или несколько номеров подтверждения (т. е. подтверждения UPDATE от партнера). Формат пакетов UPDATE определён в параграфе 5.3.5.

4.3. Обработка ошибок

Обработка ошибок HIP зависит от наличия активной ассоциации HIP. В общем случае при наличии ассоциации HIP между отправителем и получателем пакета, вызвавшего ошибку, получателю **следует** отвечать пакетом NOTIFY. Если же ассоциации нет или получатель не может надёжно определить отправителя, он **может** ответить подходящим сообщением ICMP (см. параграф 5.4).

Протокол HIP и его конечный автомат разработаны так, чтобы обеспечивать восстановление после сбоя и потери состояния одной из сторон. Ниже указаны основные варианты обработки ошибок.

Нет предыдущего состояния между системами.

Данные передаёт Инициатор. Процесс является стандартным базовым обменом 4 для создании ассоциации HIP.

Система с данными для передачи не имеет состояния с получателем, но у того остается ассоциация HIP.

Данные передаёт Инициатор, который действует как при отсутствии предшествующего состояния, передавая пакет I1 и получая R1. Когда Ответчик получает действительный пакет I2, старая ассоциация «обнаруживается» и удаляется, а вместо неё создаётся новая.

Система с данными для передачи имеет ассоциацию HIP, но ее нет у получателя.

Система передаёт данные в исходящую защищённую пользовательскую ассоциацию. Получатель «обнаруживает» ситуацию, когда он получил пакет с данными пользователя, но не может связать его с ассоциацией HIP. Принимающий хост **должен** отбросить пакет.

Принимающему хосту **следует** передать пакет ICMP типа Parameter Problem для информирования отправителя об отсутствии ассоциации HIP (см. параграф 5.4) и он **может** инициировать новый обмен HIP BEX. Однако использование этих дополнительных механизмов зависит от реализации и политики. Если передающее приложение не ждёт отклика, система может передать в этом состоянии большое число пакетов, поэтому **рекомендуется** отправка 1 или нескольких пакетов ICMP. Однако частота таких откликов **должна** быть ограничена для предотвращения злоупотреблений (см. параграф 5.4).

4.4. Конечный автомат HIP

Сам протокол HIP имеет немного состояний. В базовом обмене HIP участвуют Инициатор и Ответчик. После организации защищённой связи (security association или SA) это различие исчезает. Если состояние HIP нужно создать снова, управляющие параметры определяет партнёр, у которого сохранилось состояние и который имеет дейтаграмму для отправки другому партнёру. Описанный ниже конечный автомат пытается контролировать эти процессы.

Конечный автомат является симметричным, представляется с точки зрения Инициатора или Ответчика и не является полным представлением логики обработки. Дополнительные правила обработки приведены в описаниях пакетов, поэтому для полной реализации HIP нужна поддержка конечного автомата и этих правил.

Этот документ расширяет конечный автомат [RFC5201] и добавляет вариант перезапуска, позволяющего согласовать криптографические алгоритмы. Расширение заключается в переходе из состояния I1-SENT обратно в это же состояние (перезапуск). Инициатору требуется запускать базовый обмен HIP снова, если Ответчик не поддерживает HIT Suite Инициатора. В этом случае Инициатор перезапускает базовый обмен HIP отправкой нового пакета I1 с HIT источника, поддерживаемым Ответчиком.

Разработчикам следует понимать, что рассмотренный здесь конечный автомат является лишь описанием и логика обработки может быть реализована разными способами. В разделе 6 правила обработки пакетов рассмотрены более подробно. Это описание сосредоточено лишь на пакетах HIP I1, R1, I2 и R2. Механизмы из других спецификаций (например, многоадресность или мобильность) могут добавлять состояния и переходы в конечный автомат.

4.4.1. Термины для конечного автомата

Unused Association Lifetime (UAL) - срок действия неиспользуемой ассоциации

Определяемое реализацией время, по истечении которого хост **может** начать разрыв имеющейся ассоциации HIP при отсутствии приёма или передачи пакетов через нее.

Maximum Segment Lifetime (MSL) - максимальный срок действия сегмента

Максимальное время, которое предполагается для нахождения пакета HIP в сети. По умолчанию принято время 2 минуты, заимствованное из [RFC0793], поскольку это преобладающее допущение о сроке действия пакетов.

Exchange Complete (EC) - обмен завершен

Время, которое хост проводит в состоянии R2-SENT перед переходом в состояние ESTABLISHED. Это время составляет продолжительность тайм-аута повтора I2, умноженное на значение n приблизительно равное I2_RETRIES_MAX.

Receive ANYOTHER - получен другой пакет

Получение любого пакета, для которого не задана смена состояния или правила обработки в текущем состоянии.

4.4.2. Состояния HIP

Таблица 1. Состояния HIP.

Состояние	***Описание***
UNASSOCIATED	Запуск конечного автомата
I1-SENT	Запуск базового обмена
I2-SENT	Ожидание завершения базового обмена
R2-SENT	Ожидание завершения базового обмена
ESTABLISHED	Ассоциация HIP создана
CLOSING	Ассоциация HIP закрывается, данные передать невозможно
CLOSED	Ассоциация HIP закрыта, данные передать невозможно
E-FAILED	Отказ при базовом обмене HIP

4.4.3. Процессы состояний HIP

Поведение системы, находящейся в состоянии UNASSOCIATED, показано в таблице 2.

Таблица 2. Стартовое состояние UNASSOCIATED.

Триггер	***Действие***
Для передачи пользовательских данных нужна новая ассоциация HIP	Отправка I1 и переход в I1-SENT.
Получение I1	Передача R1 и сохранение UNASSOCIATED.
Получение I2, обработка	При успешной обработке передача R2 и переход в R2-SENT, в ином случае сохранение UNASSOCIATED.
Получение пользовательских данных для неизвестной ассоциации HIP	Необязательная передача ICMP (см. параграф 5.4) и сохранение UNASSOCIATED.
Получение CLOSE	Необязательная передача ICMP Parameter Problem и сохранение UNASSOCIATED.
Получение ANYOTHER	Отбрасывание пакета и сохранение UNASSOCIATED.

Поведение системы, находящейся в состоянии I1-SENT, показано в таблице 3.

Таблица 3. I1-SENT - инициализация базового обмена HIP.

Триггер	***Действие***
Получение I1 от Ответчика	Если локальный тег HIT меньше HIT партнера (см. параграф 6.5), пакет I1 отбрасывается, иначе передается R1. В обоих случаях сохраняется I1-SENT.
Получение I2, обработка	При успешной обработке передача R2 и переход в R2-SENT, иначе сохранение I1-SENT.
Получение R1, обработка	Если HIT Suite из локального HIT не поддерживается партнёром, выбор подходящего локального HIT, передача I1 и сохранение I1-SENT. При успешной обработке передача I2 и переход в I2-SENT, иначе сохранение I1-SENT.
Получение ANYOTHER	Отбрасывание пакета и сохранение I1-SENT.
Тайм-аут	Инкрементирование счётчика повторов. Если счётчик меньше I1_RETRIES_MAX, передача I1 и сохранение I1-SENT, если счётчик больше I1_RETRIES_MAX, переход в E-FAILED.

Поведение системы, находящейся в состоянии I2-SENT, показано в таблице 4.

Таблица 4. I2-SENT - ожидание завершения базового обмена HIP.

Триггер	***Действие***
Получение I1	Передача R1 и сохранение I2-SENT.
Получение R1, обработка	При успешной обработке передача I2. Сохранение I2-SENT

Получение I2, обработка	Если обработка успешна и локальный тег HIT меньше HIT партнера, отбрасывание I2 и сохранения I2-SENT. Если обработка успешна и локальный тег HIT больше партнеросного, передача R2 и переход в R2-SENT. При неудачной обработке сохраняется I2-SENT.
Получение R2, обработка	При успешной обработке переход в ESTABLISHED, иначе сохранение I2-SENT.
Получение CLOSE, обработка	При успешной обработке передача CLOSE_ACK и переход в CLOSED, иначе сохранение I2-SENT.
Получение ANYOTHER	Отбрасывание пакета и сохранение I2-SENT.
Тайм-аут	Инкрементирование счётчика повторов. Если счётчик меньше I2_RETRIES_MAX, передача I2 и сохранение I2-SENT, если счётчик больше I2_RETRIES_MAX, переход в E-FAILED.

Поведение системы, находящейся в состоянии R2-SENT, показано в таблице 5.

Таблица 5. R2-SENT - ожидание завершения HIP.

Триггер	***Действие***
Получение I1	Отбрасывание пакета и сохранение R2-SENT.
Получение I2, обработка	При успешной обработке передача I2. Сохранение R2-SENT
Получение R1	Отбрасывание пакета и сохранение R2-SENT.
Получение R2	Отбрасывание пакета и сохранение R2-SENT.
Получение данных или UPDATE	Переход в ESTABLISHED.
Тайм-аут завершения обмена	Переход в ESTABLISHED.
Получение CLOSE, обработка	При успешной обработке передача CLOSE_ACK и переход в CLOSED, иначе сохранение ESTABLISHED.
Получение CLOSE_ACK	Отбрасывание пакета и сохранение R2-SENT.
Получение NOTIFY	Обработка пакета и сохранение R2-SENT.

Поведение системы, находящейся в состоянии ESTABLISHED, показано в таблице 6.

Таблица 6. ESTABLISHED - ассоциация HIP создана.

Триггер	***Действие***
Получение I1	Передача R1 и сохранение ESTABLISHED.
Получение I2	Обработка головоломки и (возможно) проверка Opaque data. При успешной обработке передача R2, отбрасывание старой ассоциации HIP, создание новой и переход в R2-SENT, при отказе сохранение ESTABLISHED.
Получение R1	Отбрасывание пакета и сохранение ESTABLISHED.
Получение R2	Отбрасывание пакета и сохранение ESTABLISHED.
Получение пользовательских данных для ассоциации HIP	Обработка пакета и сохранение ESTABLISHED.
Нет пакетов в интервале UAL минут	Передача CLOSE и переход в CLOSING.
Получение UPDATE	Обработка пакета и сохранение ESTABLISHED.
Получение CLOSE, обработка	При успешной обработке send CLOSE_ACK и переход в CLOSED, при отказе сохранение ESTABLISHED.
Получение CLOSE_ACK	Отбрасывание пакета и сохранение ESTABLISHED.
Получение NOTIFY	Обработка пакета и сохранение ESTABLISHED.

Поведение системы, находящейся в состоянии CLOSING, показано в таблице 7.

Таблица 7. CLOSING - ассоциация HIP не использовалась в течение UAL минут.

Триггер	***Действие***
Для передачи данных пользователя нужна новая инкарнация ассоциации HIP	Передача I1 и переход в I1-SENT.
Получение I1	Передача R1 и сохранение CLOSING.
Получение I2, обработка	При успешной обработке передача R2 и переход в R2-SENT, иначе сохранение CLOSING.
Получение R1, обработка	При успешной обработке передача I2 и переход в I2-SENT, иначе сохранение CLOSING.
Получение CLOSE, обработка	При успешной обработке передача CLOSE_ACK, отбрасывание состояния и переход в CLOSED, иначе сохранение CLOSING.
Получение CLOSE_ACK, обработка	При успешной обработке отбрасывание состояния и переход в UNASSOCIATED, иначе сохранение CLOSING.
Получение ANYOTHER	Отбрасывание пакета и сохранение CLOSING.
Тайм-аут	Увеличение тайм-аута sum и сброс таймера. Если тайм-аут sum меньше UAL+MSL, повтор CLOSE и сохранение CLOSING. Если тайм-аут sum больше UAL+MSL, переход в UNASSOCIATED.

Поведение системы, находящейся в состоянии CLOSED, показано в таблице 8.

Таблица 8. CLOSED - передан CLOSE_ACK и при необходимости повторяется.

Триггер	***Действие***
Для передачи дейтаграммы нужна другая инкарнация ассоциации HIP	Передача I1 и сохранение CLOSED.
Получение I1	Передача R1 и сохранение CLOSED.
Получение I2, обработка	При успешной обработке передача R2 и переход в R2-SENT, иначе сохранение CLOSED.
Получение R1, обработка	При успешной обработке передача I2 и переход в I2-SENT, иначе сохранение CLOSED.
Получение CLOSE, обработка	При успешной обработке передача CLOSE_ACK. Сохранение CLOSED.
Получение CLOSE_ACK, обработка	При успешной обработке отбрасывание состояния и переход в UNASSOCIATED, иначе сохранение CLOSED.

Получение ANYOTHER	Отбрасывание пакета и сохранение CLOSED.
Тайм-аут (UAL+2MSL)	Отбрасывание состояния и переход в UNASSOCIATED

Поведение системы, находящейся в состоянии E-FAILED, показано в таблице 9.

Таблица 9. E-FAILED - отказ HIP при создании ассоциации с партнером.

Триггер	***Действие***
Ожидание в зависимости от реализации	Переход в UNASSOCIATED, после чего возможен повтор согласования.

4.4.4. Упрощённая диаграмма состояний HIP

На рисунке 2 показаны основные смены состояний. Переходы в результате приёма пакетов неявно предполагают их успешную аутентификацию или обработку.

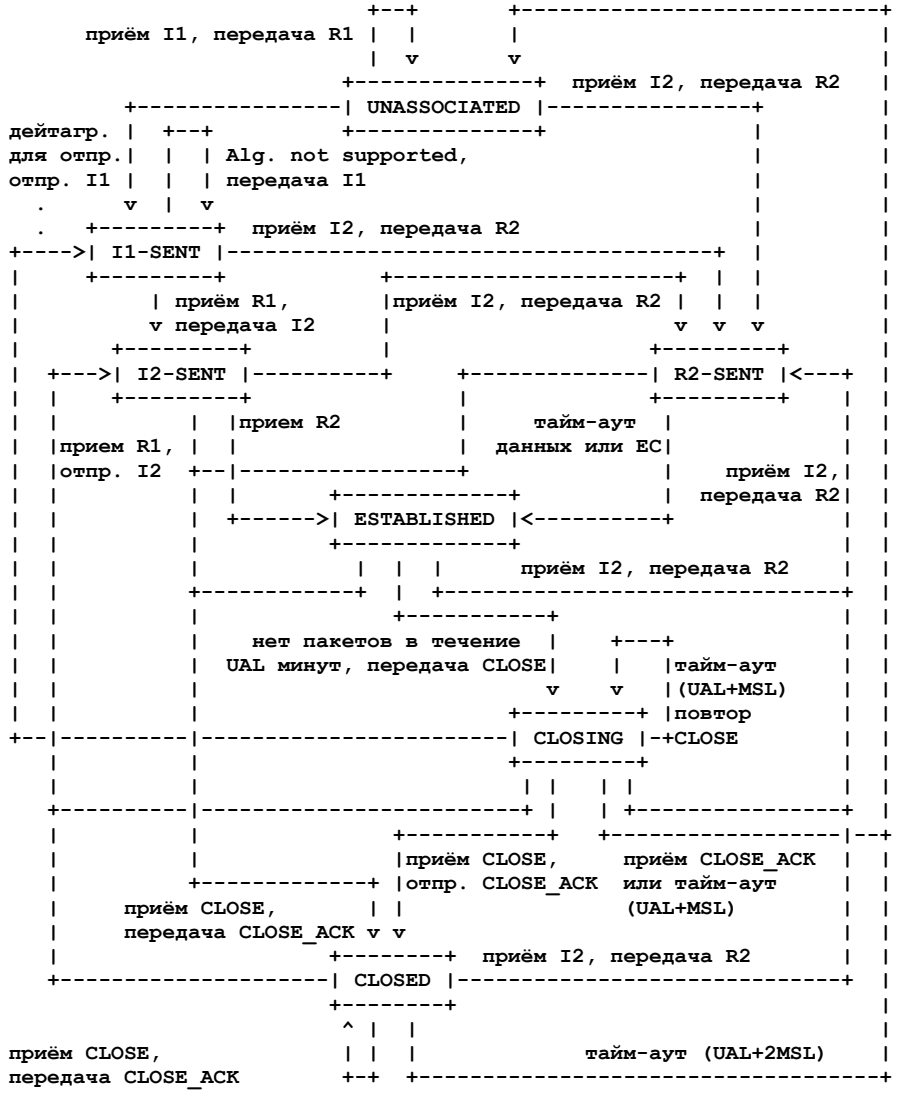


Рисунок 2. Состояния HIP.

4.5. Пользовательские данные

4.5.1. Расчёт псевдозаголовков TCP и UDP для пользовательских данных

При расчёте контрольных сумм TCP и UDP в пользовательских пакетах данных, проходящих через сокеты, привязанные к HIT, должен использоваться формат псевдозаголовка IPv6 [RFC2460], даже если фактический адрес в заголовке пакета является IPv4. Кроме того, в псевдозаголовке IPv6 **должны** использоваться теги HIT вместо адресов IPv6. Отметим, что псевдозаголовок для фактических данных HIP рассчитывается иначе (см. параграф 5.1.1).

4.5.2. Передача данных в пакетах HIP

Включение пользовательских данных в пакеты HIP могут определять другие документы, однако в настоящее время заголовок HIP является завершающим и за ним не следуют другие заголовки.

4.5.3. Транспортные форматы

Фактический формат, применяемый для передачи пользовательских данных после базового согласования HIP, этот документ не задаёт. Такие транспортные форматы и методы будут описаны в отдельных документах. Все реализации HIP **должны** поддерживать по меньшей мере транспортный формат ESP для HIP [RFC7402]. Выбор транспортного формата происходит во время базового обмена. Ответчик указывает свои предпочтения для транспорта в параметре TRANSPORT_FORMAT_LIST пакета R1, а Инициатор выбирает один из форматов и включает соответствующий параметр HIP в пакет I2.

4.5.4. Перезагрузка, тайм-аут и перезапуск HIP

Имитация потери состояния является одной из возможных DoS-атак. Ниже описан процесс управления восстановлением состояния, не открывающий возможности для DoS-атак.

Если хост перезагружен или в ассоциации HIP произошёл тайм-аут, состояние HIP теряется. Если у такого хоста сеть дейтаграмма для передачи партнёру, он просто запустит повторно базовый обмен HIP. По завершении обмена Инициатор может создать новую ассоциацию для данных (payload) и начать их отправку. Партнёр не сбрасывает своё состояние до приёма действительного пакета I2.

Если система получает пакет пользовательских данных, который не соответствует имеющейся ассоциации HIP, это может говорить о потере состояния при его сохранении у партнёра. Система **может** передать пакет ICMP типа Parameter Problem с полем Pointer, указывающим связанную с ассоциацией HIP информацию. Реакция на такие пакет зависит от реализации и среды её применения.

Если явно потерявший состояние хост хочет перезапустить базовый обмен HIP, он передаёт партнёру сообщение I1. После успешного базового обмена Инициатор может создать новую ассоциацию HIP, а партнёр отбросит свои старые ассоциации для данных и создаст новую.

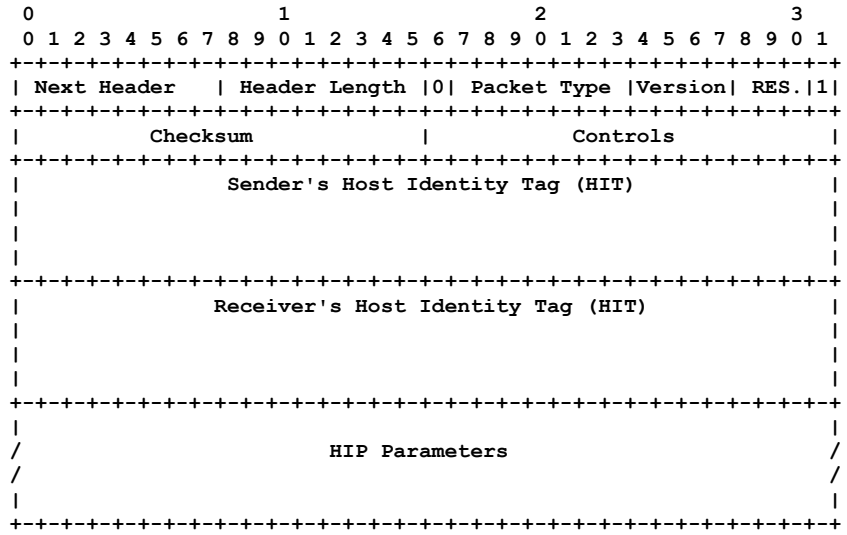
4.6. Распространение сертификатов

Этот документ не задаёт способы использования сертификатов и их передачи между хостами. Предполагается определение этих функций в будущей спецификации, как было сделано для HIP версии 1 (см. [RFC6253]). Однако значение типа параметра для переноса сертификатов зарезервировано (CERT, Type 768), см. параграф 5.2.

5. Формат пакетов

5.1. Формат данных

Все пакеты HIP начинаются с показанного ниже фиксированного заголовка.



Логически заголовок HIP является заголовком расширения IPv6, однако этот документ не описывает обработку значений Next Header, отличающихся от 59 (IPPROTO_NONE), означающего отсутствие последующих заголовков. Будущие документы **могут** задать поведение для других значений, однако текущие реализации должны игнорировать последующие данные при получении нереализованного значения Next Header.

Поле Header Length указывает общий размер HIP Header и HIP Parameters в 8-байтовых блоках без учёта первых 8 байтов. Поскольку заголовки HIP **должны** включать теги HIT отправителя и получателя, минимальное значение этого поля составляет 4, а максимальный размер поля HIP Parameters - (255 * 8) - 32 = 2008 байтов (см. параграф 5.1.3 по части фрагментации HIP). Отметим, что это задаёт дополнительное ограничение на размер параметров в поле HIP Parameters, независимо от максимального размера отдельных параметров.

Поле Packet Type указывает тип пакета HIP. Конкретные типы пакетов указаны ниже в соответствующих параграфах. При получении хостом HIP пакета HIP нераспознанного типа этот пакет **должен** отбрасываться.

Поле HIP Version занимает 4 бита, данный документ определяет версию 2. Предполагается увеличение номера версии лишь при внесении в протоколо изменений, не совместимых с прежней версией. Большинство расширений может обрабатываться за счёт определения новых типов пакетов, параметров или элементов управления Control (см. параграф 5.1.2). Три следующих бита зарезервированы на будущее и **должны** сбрасываться в 0 при передаче, а получатель **должен** игнорировать их.

Два фиксированных бита в заголовке зарезервированы для совместимости с SHIM6 параграфом 5.3 в [RFC5533]. В реализациях, следующих лишь этой спецификации, эти биты **должны** устанавливаться, как показано на рисунке, а получатели **должны** игнорировать их. Это обеспечит оптимальную совместимость на будущее. Отметим, что реализации, которые следуют другим совместимым спецификациям в дополнение к этой, правила установки этих битов могут быть иными. Например, реализации, поддерживающей данную спецификацию и протокол SHIM6, нужно проверять эти биты для определения способа обработки пакета.

Поля HIT всегда имеют размер 128 битов (16 байтов).

5.1.1. Контрольная сумма

Поскольку контрольная сумма учитывает поля адресов отправителя и получателя в заголовке IP, она **должна** пересчитываться понимающими HIP устройствами NAT.

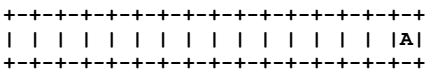
Если для пакета HIP используется протокол IPv6, псевдозаголовок [RFC2460] содержит адреса отправителя и получателя IPv6, размер пакета HIP в поле Length псевдозаголовка, нулевое поле и номер протокола HIP (см. параграф 5.1) в поле Next Header. Поле Length указывает размер в байтах и может быть рассчитано из поля HIP Header Length

```
(HIP Header Length + 1) * 8
```

В случае IPv4 используется формат псевдозаголовка IPv4 UDP [RFC0768]. В псевдозаголовке используются адреса отправителя и получателя из заголовка IP, нулевое поле, номер протокола HIP (см. раздел 4) и размер, определяемый как для IPv6.

5.1.2. Поле HIP Controls

Поле HIP Controls переносит информацию о структуре пакета и созможностях хоста. В настоящее время определён лишь 1 бит.



A - Anonymous

Установка этого бита говорит о том, что HI отправителя этого пакета является анонимным, т. е. не включён в каталог. Сохранять анонимные HI **не следует**. Партнёр, получивший анонимный HI в пакете R1 или I2, может отвергнуть этот пакет.

Остальные биты зарезервированы на будущее и **должны** сбрасываться в 0 при передаче, а получатель **должен** игнорировать их.

5.1.3. Поддержка фрагментации HIP

Реализация HIP **должна** поддерживать фрагментацию и сборку IP. Сборка фрагментов **должна** быть реализована для IPv4 и IPv6, а фрагментацию **требуется** реализовать для IPv4 (стек и сети IPv4 обычно делают это по умолчанию) и **рекомендуется** для IPv6. В сетях IPv6 минимальное значение MTU (1280 байтов) больше чем в IPv4. Большего размера MTU обычно достаточно для большинства пакетов HIP, поэтому фрагментация может не потребоваться. Если предполагается передача пакетов HIP больше минимального IPv6 MTU, реализация **должна** поддерживать создание фрагментов даже для IPv6.

В сетях IPv4 пакеты HIP могут сталкиваться с малым значением MTU на пути передачи. Поскольку базовый протокол HIP, заданный этим документов, не включает механизма использования нескольких дейтаграмм IP для одного пакета HIP, поддержка обнаружения MTU на пути не имеет значения для HIP в сетях IPv4. Понимающим HIP устройствам NAT **следует** выполнять фрагментацию и сборку для пакетов HIP.

Реализации HIP должны соблюдать осторожность в алгоритмах сборки фрагментов для устойчивости к DoS-атакам.

Использование цепочек сертификатов может приводить к фрагментации пакетов, а та может открывать возможность для DoS-атак [KAU03]. Можно применять схему «хэш и URL», определённую в [RFC6253] для HIP версии 1, чтобы избежать фрагментации и смягчить DoS-атаки.

5.2. Параметры HIP

Параметры HIP содержат сведения, требуемые для организации и поддержки ассоциаций HIP. Например, это могут быть открытые ключи партнёра и сигналы согласования шифров и обработки данных, а также дополнительная информация, важная для конечных хостов и промежуточных устройств. Спецификация параметров HIP и их сопоставления с пакетами HIP и типами пакетов достаточно гиби и позволяют расширениям HIP определять новые параметры и поведение протокола. В пакетах HIP параметры упорядочиваются по численным значениям их типа и представляются в формате TLV. Определённые данным документом параметры приведены в таблице.

TLV	*Тип*	*Размер*	*Данные*
R1_COUNTER	129	12	Счётчик генерации Puzzle.
PUZZLE	257	12	#K и Random #I.
SOLUTION	321	20	#K, Random #I и решение головоломки #J.
SEQ	385	4	Идентификатор пакета UPDATE.
ACK	449	переменный	Идентификатор пакета UPDATE.
DH_GROUP_LIST	511	переменный	Упорядоченный список DH Group ID, поддерживаемых хостом.
DIFFIE_HELLMAN	513	переменный	Открытый ключ.
HIP_CIPHER	579	переменный	Список алгоритмов шифрования HIP.
ENCRYPTED	641	переменный	Шифрованная часть пакета HIP.
HOST_ID	705	переменный	Host Identity с FQDN или NAI.
HIT_SUITE_LIST	715	переменный	Упорядоченный список HIT Suite, поддерживаемых Ответчиком.
CERT	768	переменный	Сертификат HI (задан в отдельном документе).
NOTIFICATION	832	переменный	Информационные сведения.
ECHO_REQUEST_SIGNED	897	переменный	Подписанные Opaque data для возврата
ECHO_RESPONSE_SIGNED	961	переменный	Подписанные Opaque data, возвращаемые по запросу.
TRANSPORT_FORMAT_LIST	2049	переменный	Упорядоченный список предпочтительных номеров типов транспорта HIP.
HIP_MAC	61505	переменный	Код аутентификации сообщения HMAC с ключевым материалом из KEYMAT.
HIP_MAC_2	61569	переменный	Код аутентификации сообщения HMAC с ключевым материалом из KEYMAT. В отличие от HIP_MAC, в расчёт HIP_MAC_2 включается параметр HOST_ID.
HIP_SIGNATURE_2	61633	переменный	Подпись, используемая в пакете R1.

HIP_SIGNATURE	61697	переменный	Подпись, используемая в пакете R1.
ECHO_REQUEST_UNSIGNED	63661	переменный	Opaque data для возврата после подписи
ECHO_RESPONSE_UNSIGNED	63425	переменный	Opaque data, возвращаемые по запросу, после подписи.

Поскольку порядок (по возрастанию) параметров HIP строго соблюдается (см. параграф 5.2.1), выделенные значения параметров разделены промежутками для будущих назначений. Диапазоны параметров показаны в таблице.

Диапазон типов	***Назначение***
0 - 1023	Согласование.
1024 - 2047	Резерв.
2048 - 4095	Параметры, относящиеся к транспортным форматам HIP.
4096 - 8191	Подписываемые параметры, выделенные соответствующими спецификациями.
8192 - 32767	Резерв.
32768 - 49151	Подписываемые параметры. Резерв для приватного использования.
49152 - 61439	Резерв.
61440 - 62463	Подписи и (подписанные) MAC.
62464 - 63487	Параметры, для которых не применяются подписи и MAC.
63488 - 64511	Rendezvous и ретрансляция.
64512 - 65023	Параметры, для которых не применяются подписи и MAC.
65024 - 65535	Резерв.

Процедура определения новых параметров описана в параграфе 5.2.2. Диапазон от 32768 (215) до 49151 (215 + 214) зарезервирован для приватного использования (Reserved for Private Use) и типы из него **следует** выбирать случайным образом для снижения вероятности конфликтов.

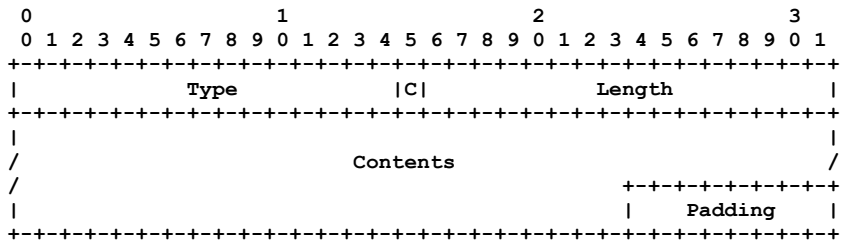
5.2.1. Формат TLV

В следующих параграфах описываются параметры, представляемые в формате TLV. Значение поля Type в этих параметрах определяет порядок параметров в пакете. Параметры **должны** включаться в пакет строго в порядке возрастания значений типа. При наличии в пакете нескольких однотипных параметров они **должны** размещаться в пакете один за другим. При нарушении порядка параметров формат пакета считается некорректным и пакет **должен** отбрасываться.

Параметры типов 2048 - 4095 относятся к транспортному формату. В настоящее время определён лишь формат ESP [RFC7402].

Все параметры TLV имеют размер (с учётом полей Type и Length), кратный 8 байтам. При необходимости в параметр **должно** добавляться заполнение до кратного 8 байтам размера. Это правило обеспечивает корректное выравнивание данных. В байтах заполнения отправитель **должен** устанавливать значение 0, а получателю **не следует** проверять эти значения.

Поле Length указывает размер поля Contents (в байтах), поэтому общий размер параметра TLV (поля Type, Length, Contents, Padding) связан со значением поля Length выражением 11 + Length - (Length + 3) % 8, где % указывает деление по модулю.



Type

Код типа параметра размером 16 битов, бит C является частью поля Type.

C

Бит критичности параметра. Устанавливается (1) для параметров, которые получатель **должен** распознавать, и сбрасывается в остальных случаях. Бит C является частью поля Type, поэтому критичные параметры всегда имеют нечетное значение типа, некритичные - четное.

Length

Размер поля Contents в байтах. Поля Type, Length, Padding не учитываются.

Contents

Зависит от типа параметра.

Padding

Заполнение до 7 байтов при необходимости.

Критические параметры (нечетное значение типа) **должны** распознаваться получателем. Если получатель встречает неизвестные критический параметр, дальнейшая обработка пакета **недопустима**. Получатель **может** передать сообщение ICMP или пакет NOTIFY, как указано в параграфе 4.3. Некритические параметры **можно** игнорировать и получателю, встретившему такой неизвестный параметр, **следует** продолжать обработку пакета, как будто параметра нет.

5.2.2. Определение новых параметров

Новые параметры при необходимости могут быть определены в новых спецификациях. При определении новых параметров следует соблюдать осторожность, чтобы номера типов были подходящими и оставалось место для будущих расширений. Нужно помнить, что параметры **должны** упорядочиваться по росту значений поля Type (см. параграф 5.2.1). При определении новых параметров **должны** соблюдаться приведённые ниже правила.

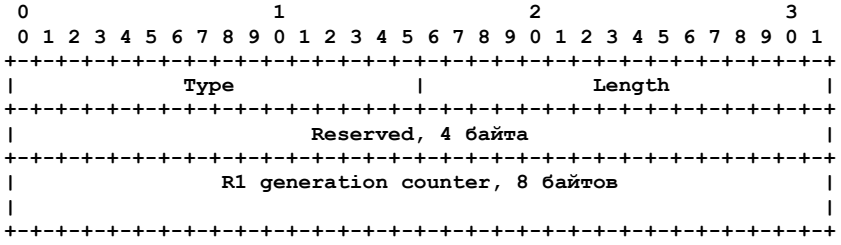
1. Младший бит (C) в коде типа служит для указания критических параметров. Поэтому типы с нечетным номером всегда задают критические параметры.

2. Новый параметр **может** быть критическим лишь в том случае, когда старая реализация при его игнорировании создаст проблему безопасности. В общем случае новые параметры **следует** задавать как некритические и ожидать для них отклика от получателя.

3. Система, реализующая новый критический параметр, **должна** обеспечивать возможность отключить соответствующее свойство, чтобы критический параметр не передавался. Параметр конфигурации **должен** быть документирован. Реализация, работающая в режиме соответствия данной спецификации, **должна** по умолчанию отключать передачу критических параметров. Иными словами, интерфейс управления **должен** разрешать обычный режим с возможностью позволять отключать и включать новые критические параметры.

4. Правила выделения кодов Type приведены в разделе 9.

5.2.3. R1_COUNTER

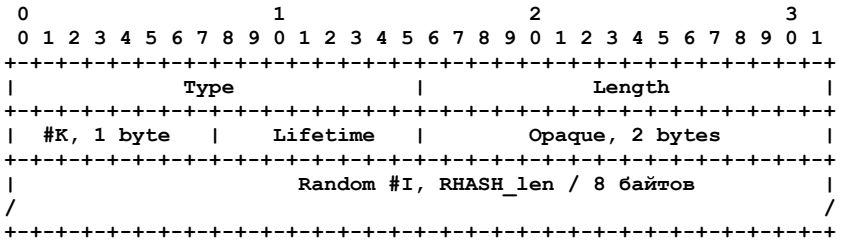


Type
129
Length
12
R1 generation counter
Текущее поколение действительных головоломок.

Параметр R1_COUNTER содержит 64-битовое целое число без знака с сетевым порядком байтов, указывающее текущее поколение действительных головоломок. Отправителю **следует** периодически инкрементировать этот счётчик. **рекомендуется** инкрементировать счётчик не реже отмены старых значений PUZZLE, чтобы значения SOLUTION для них больше не принимались.

Поддержка параметра R1_COUNTER является обязательной, хотя его включение в пакет R1 не требуется. Его **следует** включать в R1 (тогда оно учитывается в подписи) и при наличии параметра в R1 Инициатор **должен** возвращать его (включая поле Reserved) в пакете I2.

5.2.4. PUZZLE



Type
257
Length
4 + RHASH_len / 8
#K
Число проверяемых битов.
Lifetime
Срок действия головоломки - $2^{(Lifetime - 32)}$ секунд.
Opaque
Данные, установленные Ответчиком для индексирования головоломки.
Random #I
Случайное значение размером RHASH_len битов.

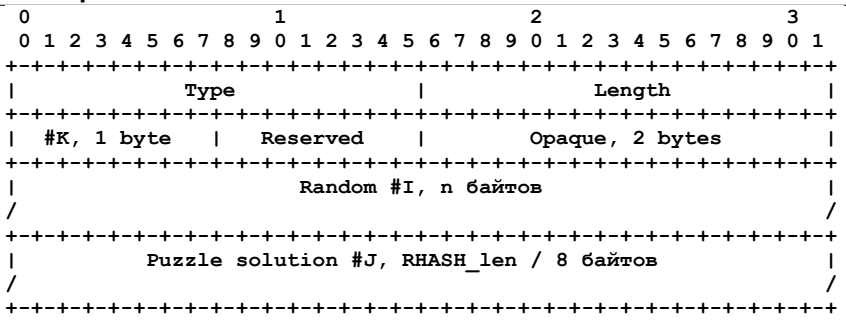
Random #I представляется целым числом размером n битов (n = RHASH_len), а #K и Lifetime - 8-битовыми целыми числами с сетевым порядком байтов.

Параметр PUZZLE указывает сложность головоломки #K и n-битовое случайное целое число Random #I. Поле Lifetime указывает время, в течение которого действует решение головоломки, и ограничивает время, которое Инициатор может потратить на решение задачи. Срок действия указывается степенью 2 и фактический срок составляет $2^{(Lifetime - 32)}$ секунд. Задача **может** быть дополнена параметром ECHO_REQUEST_SIGNED или ECHO_REQUEST_UNSIGNED, включаемым в пакет R1. Содержимое запроса возвращается в параметре ECHO_RESPONSE_SIGNED или ECHO_RESPONSE_UNSIGNED, позволяя Ответчику использовать значение как часть обработки головоломки.

Поля Opaque и Random #I не учитываются в параметре HIP_SIGNATURE_2.

5.2.5. SOLUTION

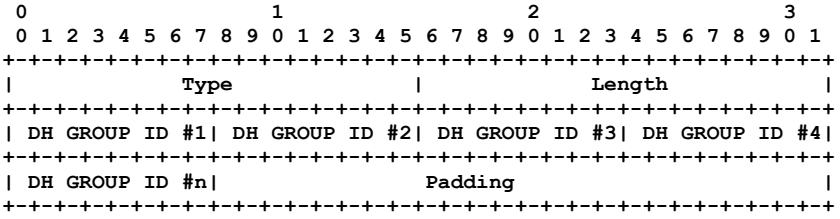
Type
321
Length
4 + RHASH_len / 4
#K
Число проверяемых битов.



Reserved
0 при передаче, игнорируется при получении.
Opaque
Копируется без изменений из полученного параметра PUZZLE.
Random #I
Случайное значение размером RHASH_len битов.
Puzzle solution #J
Случайное значение размером RHASH_len битов.
Random #I и Random #J представляются целами числами без знака размером n битов (n = RHASH_len), а #K - 8-битовое целое число без знака. Все числа используют сетевой порядок байтов.

Параметр SOLUTION содержит решение головоломки. Он возвращает случайное значение сложности #K, поле Opaque и целочисленное представление головоломки #I.

5.2.6. DH_GROUP_LIST

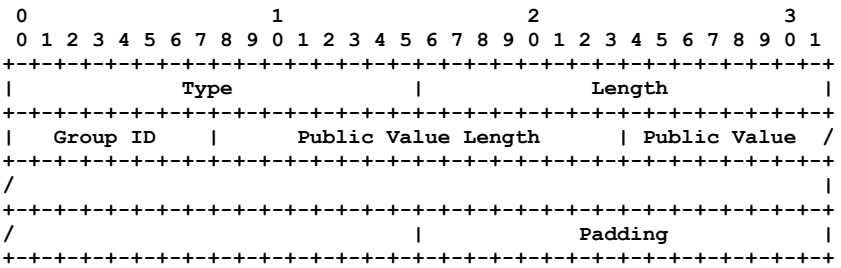


Type
511
Length
Число DH Group ID.
DH GROUP ID
Идентификатор группы DH, поддерживаемой хостом. Список ID упорядочивается по предпочтениям хоста. Возможные DH Group ID указываются в параметре DIFFIE_HELLMAN. Размер DH Group ID составляет 1 октет.
Параметр DH_GROUP_LIST содержит список поддерживаемых хостом DH Group ID. Инициатор передаёт DH_GROUP_LIST в пакете I1, а Ответчик Responder передаёт свой список в подписанной чати пакета R1. DH Group ID в DH_GROUP_LIST указываются в порядке снижения предпочтений хоста, передающего список. Информация в DH_GROUP_LIST позволяет Ответчику выбрать предпочтительную группу DH, поддерживаемую Инициатором. На основе DH_GROUP_LIST в пакете R1 Инициатор может определить, выбрал ли Ответчик наилучший вариант из числа предпочитается обеими сторонами. Если Ответчик выбрал не лучший вариант, Инициатор может считать это попыткой атаки на понижение (см. параграф 4.1.7).

При выборе группы DH для параметра DIFFIE_HELLMAN в пакете R1 Ответчик **должен** выбрать первый идентификатор DH Group ID из своего DH_GROUP_LIST в пакете R1, который совместим с одним из Suite ID списке Инициатора DH_GROUP_LIST из пакета I1. Ответчику **недопустимо** выбирать какой-либо иной DH Group ID из числа содержащихся в обоих списках во избежание незамеченной атаки на понижение.

В общем случае хостам **следует** предпочитать более сильные группы, если сложность расчётов не слишком велика для приложения.

5.2.7. DIFFIE_HELLMAN



Type
513
Length
Размер в октетах без учёта Type, Length, Padding.
Group ID
Указывает значения для p и g, а также KDF.
Public Value Length
Размер следующего поля Public Value в октетах.

Public Value

Открытый ключ DH отправителя.

Один параметр DIFFIE_HELLMAN может быть включён в выбранный пакет HIP на основе выбранного идентификатора DH Group ID (параграф 5.2.6). Ниже перечислены определённые в настоящее время Group ID со значениями.

Группа	*KDF*	*Значение*
Резерв		0
Отменена		1
Отменена		2
1536-bit MODP group [RFC3526]	HKDF [RFC5869]	3
3072-bit MODP group [RFC3526]	HKDF [RFC5869]	4
Отменена		5
Отменена		6
NIST P-256 [RFC5903]	HKDF [RFC5869]	7
NIST P-384 [RFC5903]	HKDF [RFC5869]	8
NIST P-521 [RFC5903]	HKDF [RFC5869]	9
SECP160R1 [SECG]	HKDF [RFC5869]	10
2048-битовая группа MODP [RFC3526]	HKDF [RFC5869]	11

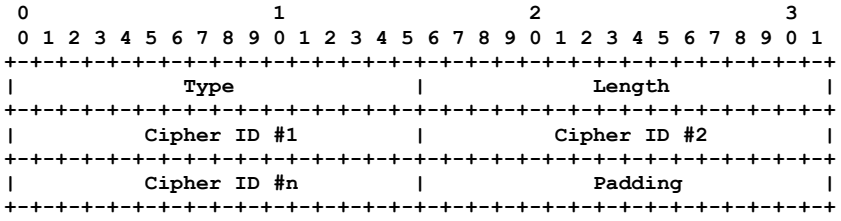
Группы MODP DH определены в [RFC3526], группы ECDH 7-9 - в [RFC5903] и [RFC6090], ECDH 10 описана в Приложении D. Любые группы ECDH, используемые с HIP, **должны** иметь коэффициент (co-factor) 1.

Group ID указывает также функцию вывода ключей для создания симметричных ключей для HMAC и симметричного шифрования из обмена ключами DH (см. параграф 6.5).

Реализации HIP **должны** поддерживать Group ID 3. 160-битовая группа SECP160R1 может применяться при невысоких требованиях к защите (например, для просмотра web) или при ограниченных возможностях оборудования (например, в некоторых PDA). Разработчикам **следует** реализовать Group ID 4 и 8.

Для предотвращения ненужных отказов в процессе базового обмена остальные группы **следует** реализовать на хостах с достаточными ресурсами.

5.2.8. HIP_CIPHER



Type

579

Length

Размер в октетах без учёта Type, Length, Padding.

Cipher ID

Идентификатор алгоритма, используемого для шифрования параметра ENCRYPTED. Ниже перечислены определённые в настоящее время Cipher ID.

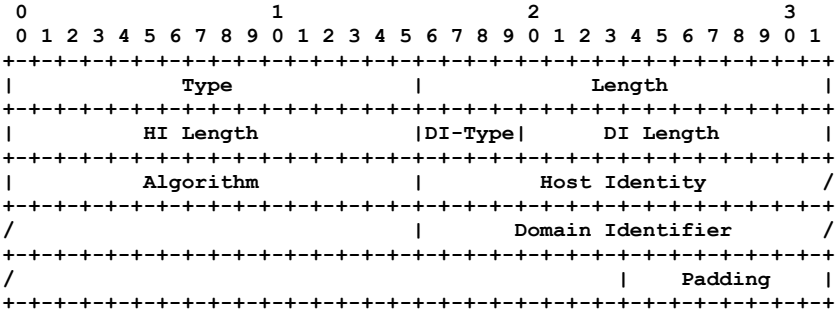
Suite ID	*Значение*
Резерв	0
NULL-ENCRYPT	1 ([RFC2410])
AES-128-CBC	2 ([RFC3602])
Резерв	3 (не используется)
AES-256-CBC	4 ([RFC3602])

Отправитель параметра HIP_CIPHER **должен** обеспечить в нем не более 6 полей Cipher ID, а получатель **должен** быть готов обрабатывать параметры, содержащие более 6 полей Cipher ID, воспринимая из них 6 первых и отбрасывая остальные. Ограничение числа Cipher ID определяет максимальный размер параметра HIP_CIPHER. По умолчанию параметр HIP_CIPHER **должен** включать хотя бы один обязательный Cipher ID. **Может** применяться параметр конфигурации для переопределения администратором заданного по умолчанию идентификатора.

Ответчик указывает поддерживаемые и желаемые Cipher ID (до 6) в порядке предпочтения в пакете R1, а Инициатор **должен** выбрать из них лишь один Cipher ID, который применяется для генерации параметра ENCRYPTED.

Обязательная реализация - AES-128-CBC. Разработчикам **следует** поддерживать NULL-ENCRYPT для тестирования и отладки, но **недопустимо** предлагать или воспринимать его, если явно не задано тестирование или отладка HIP.

5.2.9. HOST_ID



Type
705
Length
Размер в октетах без учёта Type, Length, Padding.
HI Length
Размер поля Host Identity в октетах.
DI-Type
Тип следующего поля Domain Identifier.
DI Length
Размер поля Domain Identifier в октетах.
Algorithm
Индекс используемого алгоритма.
Host Identity
Фактическое значение Host Identity.
Domain Identifier
Идентификатор отправителя.
Определённые в настоящее время значения DI-Type приведены в таблице.

Тип	***Значение***
Не включено	0
FQDN	1
NAI	2

FQDN
Полное доменное имя (Fully Qualified Domain Name) в двоичном формате.
NAI
Идентификатор доступа в сеть (Network Access Identifier).
Формат FQDN определён в параграфе 3.1 RFC 1035 [RFC1035], формат NAI - в [RFC4282].

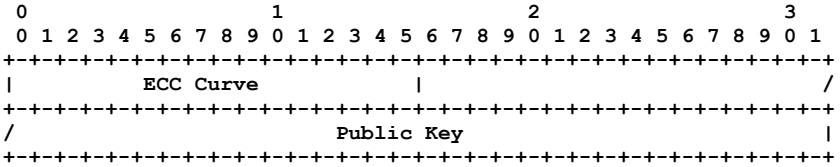
Хост **может** связать Host Identity с одним Domain Identifier в параметре HOST_I. При отсутствии Domain Identifier (DI-Type = 0), в поле DI Length также устанавливается 0.

Определённые в настоящее время HI Algorithm перечислены ниже.

Профили алгоритмов	***Значения***
Резерв	0
DSA	3 [FIPS.186-4.2013] (**рекомендуется**)
RSA	5 [RFC3447] (**требуется**)
ECDSA	7 [RFC4754] (**требуется**)
ECDSA_LOW	9 [SECG] (**рекомендуется**)

Для ключей DSA, RSA, ECDSA следует применять профили, содержащие по меньшей мере 112 битов «силы защиты» (как определено в [NIST.800-131A.2011]). Для дополнения подписи RSA **должно** применяться заполнение по методу вероятностной схемы подписи (Probabilistic Signature Scheme или PSS) [RFC3447].

Host Identity выводится из формата DNSKEY для RSA и DSA. Для этого применяется поле Public Key части RDATA из RFC 4034 [RFC4034]. Для криптографии на основе эллиптических кривых (Elliptic Curve Cryptography или ECC) выделены два профиля - ECDSA и ECDSA_LOW. ECC содержит кривые, одобренные NIST и определённые в RFC 4754 [RFC4754]. Профиль ECDSA_LOW определён для устройств с ограниченными вычислительными возможностями и использует более короткие кривые из Standards for Efficient Cryptography Group [SECG]. При использовании ECDSA с HIP **должен** применяться коэффициент 1.



Для ECDSA и ECDSA_LOW отождествление хоста (Host Identity) представляется двумя полями.

ECC Curve
Метка кривой.
Public Key
Открытый ключ в форме строки октетов Represented [RFC6090].
Для хостов, применяющих ECDSA в качестве алгоритма, требуются указанные ниже кривые ECC.

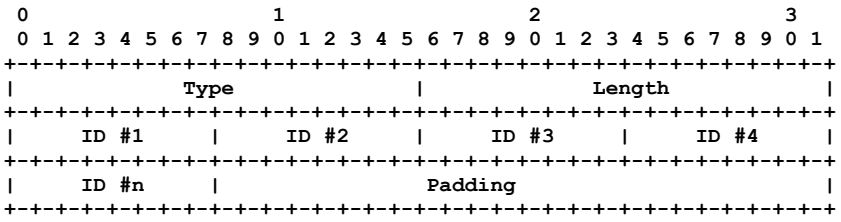
Алгоритм	***Кривая***	***Значения***
ECDSA	Резерв	0
ECDSA	NIST P-256	1 [RFC4754]
ECDSA	NIST P-384	2 [RFC4754]

Для хостов, реализующих профиль алгоритма ECDSA_LOW, требуются указанные ниже кривые.

Алгоритм	*Кривая*	*Значения*
ECDSA_LOW	Резерв	0
ECDSA_LOW	SECP160R1	1 [SECG]

5.2.10. HIT_SUITE_LIST

Параметр HIT_SUITE_LIST содержит список поддерживаемых Ответчиком HIT Suite ID. Ответчик передает HIT_SUITE_LIST в подписанной части пакета R1. На основе HIT_SUITE_LIST Инициатор может определить, какие наборы из исходного списка HIT Suite ID поддерживает Ответчик.



Type

715

Length

Число HIT Suite ID

ID

Однооктетный идентификатор HIT Suite, поддерживаемого хостом. Список идентификаторов упорядочивается по уровню предпочтения. 4 старших бита поля ID соответствуют HIT Suite ID в поле ORCHID OGA ID, а младшие биты являются резервными и устанавливаются отправителем в 0. Получение ID с отличными от 0 четырьмя младшими битами **следует** считать ошибкой, которая **может** вести к отправке NOTIFICATION типа UNSUPPORTED_HIT_SUITE.

HIT Suite ID индексирует HIT Suite, включающие алгоритмы подписи, как указано в параграфе 5.2.9, и хэш-функции.

Поле ID в HIT_SUITE_LIST является 8-битовым в отличие от 4-битовых HIT Suite ID и OGA ID в ORCHID. Это предназначено для расширения пространства HIT Suite ID, если 16 доступных значений будет недостаточно. В этом случае значение 0 (одно из 16) указывает, что 4 дополнительных бита ORCHID будут служить для кодирования HIT Suite ID. Текущие 4-битовые HIT Suite ID используют лишь старшие биты поля I, а в будущих документах может быть определено использование 4 младших битов поля ID.

Ниже перечислены определённые в настоящее время HIT Suite ID и связи между 4-битовыми ID, применяемыми в поле OGA ID, и 8-битовым представлением в HIT_SUITE_LIST ID.

HIT Suite	*4-битовый идентификатор*	*8-битовое кодирование*
Резерв	0	0x00
RSA,DSA/SHA-256	1	0x10 (**требуется**)
ECDSA/SHA-384	2	0x20 (**рекомендуется**)
ECDSA_LOW/SHA-1	3	0x30 (**рекомендуется**)

В таблице 10 комбинации HIT Suite представлены более подробно. Каждый алгоритм генерации принимает на входе идентификатор HI, как указано в параграфе 3.2. Результат размером 96 битов напрямую используется в ORCHID.

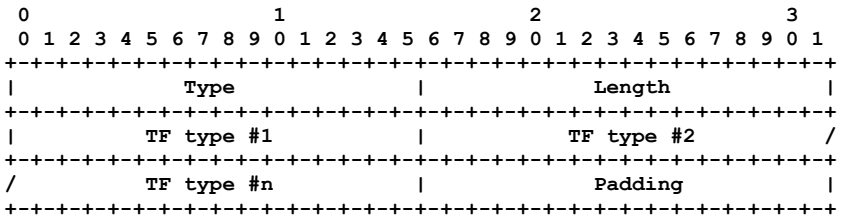
Таблица 10. Наборы HIT Suite.

Индекс	*Хэш-функция*	*HMAC*	*Семейство алгоритмов подписи*	*Описание*
0				Резерв
1	SHA-256	HMAC-SHA-256	RSA, DSA	RSA или DSA HI хешируется SHA-256 и отсекается до 96 битов.
2	SHA-384	HMAC-SHA-384	ECDSA	ECDSA HI хешируется SHA-384 и отсекается до 96 битов.
3	SHA-1	HMAC-SHA-1	ECDSA_LOW	ECDSA_LOW HI хешируется SHA-1 и отсекается до 96 битов.

Хэш Ответчика, как указано в HIT Suite, определяет код HMAC, применяемый для функции RHASH. В настоящее время определены кода HMAC-SHA-256 [RFC4868], HMAC-SHA-384 [RFC4868], HMAC-SHA-1 [RFC2404].

5.2.11. TRANSPORT_FORMAT_LIST

Параметр TRANSPORT_FORMAT_LIST содержит список поддерживаемых Ответчиком форматов транспорта HIP (TF) и передаётся в подписанной части пакета R1. На основе TRANSPORT_FORMAT_LIST Инициатор выбирает 1 подходящий транспорт в своём пакете отклика.



Type

2049

Length

Удвоенное число типов TF.

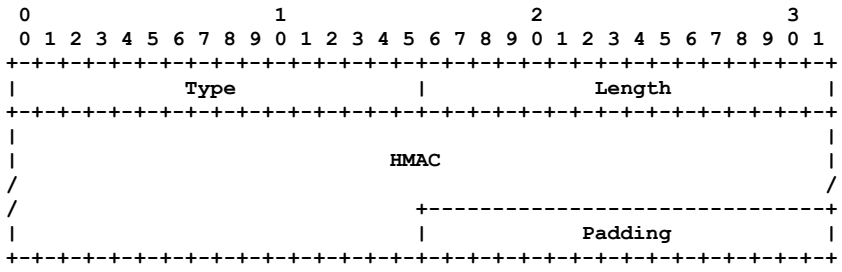
TF Type

Тип транспортного формата (TF), поддерживаемого хостом. Номера типов соответствуют номерам параметров HIP для соответствующего транспорта. Список TF упорядочивается по предпочтениям отправителя.

Типы TF индексируют соответствующие параметры HIP для формата транспорта из диапазона 2050 - 4095. Параметры и их применение задают отдельные документы. В настоящее время определён лишь транспорт IPsec ESP [RFC7402].

Для каждого указанного типа TF отправитель TRANSPORT_FORMAT_LIST **должен** включить в пакет HIP соответствующий параметр формата транспорта. Получатель **должен** игнорировать тип TF в TRANSPORT_FORMAT_LIST при отсутствии в пакете соответствующего параметра транспортного формата.

5.2.12. HIP_MAC



Type
61505

Length
Размер в октетах без учёта Type, Length, Padding.

HMAC
Код HMAC рассчитанный для пакета HIP без учёта HIP_MAC и последующих параметров, таких как HIP_SIGNATURE, HIP_SIGNATURE_2, ECHO_REQUEST_UNSIGNED, ECHO_RESPONSE_UNSIGNED. В поле Checksum **должно** устанавливаться значение 0, а поле HIP Header Length в базовом заголовке HIP **должно** рассчитываться без учёта исплюченных при вычислении HMAC параметров. Размер HMAC является естественным размером вывода применяемой хэш-функции.

HMAC использует RHASH как алгоритм хэширования. Расчёт и проверка описаны в параграфе 6.4.1.

5.2.13. HIP_MAC_2

HIP_MAC_2 - это код MAC пакета и HI отправителя в форме параметра HOST_ID, когда этот параметр фактически в пакет не включен. Структура параметра описана в параграфе 5.2.12.

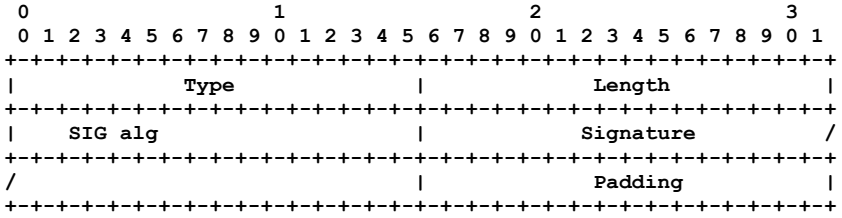
Type
61569

Length
Размер в октетах без учёта Type, Length, Padding.

HMAC
Код HMAC рассчитанный для пакета HIP без учёта HIP_MAC_2 и последующих параметров, таких как HIP_SIGNATURE, HIP_SIGNATURE_2, ECHO_REQUEST_UNSIGNED, ECHO_RESPONSE_UNSIGNED. В поле Checksum **должно** устанавливаться значение 0, а поле HIP Header Length в базовом заголовке HIP **должно** рассчитываться без учёта исплюченных при вычислении HMAC параметров. Размер HMAC является естественным размером вывода применяемой хэш-функции.

HMAC использует RHASH как алгоритм хэширования. Расчёт и проверка описаны в параграфе 6.4.1.

5.2.14. HIP_SIGNATURE



Type
61697

Length
Размер в октетах без учёта Type, Length, Padding.

SIG alg
Алгоритм подписи.

Signature
Подпись рассчитывается для пакета HIP без параметра HIP_SIGNATURE и следующих за ним параметров. При расчёте подписи для поля Checksum **должно** устанавливаться значение 0, а поле HIP Header Length в базовом заголовке HIP **должно** рассчитываться лишь до начала параметра HIP_SIGNATURE.

Алгоритм подписи определён в параграфе 5.2.9. Поле Signature кодируется в зависимости от применяемого алгоритма (например, в соответствии с [RFC3110] для RSA/SHA-1, [RFC5702] для RSA/SHA-256, [RFC2536] для DSA, [RFC6090] для ECDSA). Расчёт и проверка HIP_SIGNATURE описаны в параграфе 6.4.2.

5.2.15. HIP_SIGNATURE_2

HIP_SIGNATURE_2 исключает переменные параметры пакета R1 для создания R1 заранее. Структура параметра описана в параграфе 5.2.14.

Type
61633

Length
Размер в октетах без учёта Type, Length, Padding.

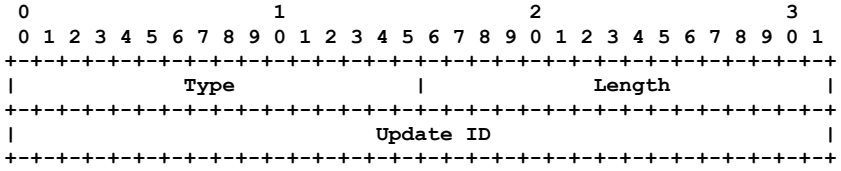
SIG alg
Алгоритм подписи.

Signature

В пакете R1 с параметром HIP_SIGNATURE_2 поля HIT Инициатора, Checksum, а также Opaque и Random #I в PUZZLE **должны** иметь значение 0 при расчёте подписи HIP_SIGNATURE_2. Поле HIP Header Length в базовом заголовке HIP **должно** быть скорректировано, как будто при расчёте подписи параметра HIP_SIGNATURE_2 не было в пакете, т. е. HIP Header Length указывает при проверке и подписывании начало HIP_SIGNATURE_2.

Обнуление HIT Инициатора позволяет создать пакеты R1 заранее для минимизации последствий возможных DoS-атак. Обнуление Random #I и Opaque в параметре PUZZLE позволяет динамически заполнять эти поля в заранее созданных R1. Расчёт и проверка HIP_SIGNATURE_2 описаны в параграфе 6.4.2.

5.2.16. SEQ



Type

385

Length

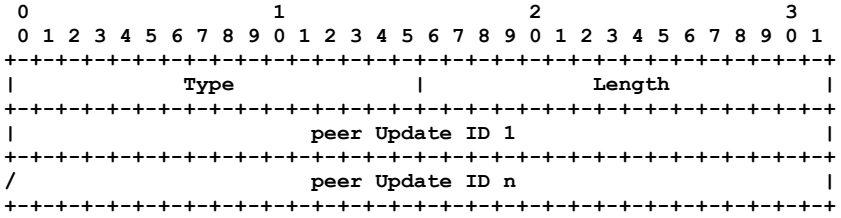
4

Update ID

32-битовый порядковый номер.

Update ID является целым числом без знака с сетевым порядком байтов и хост устанавливает для него значение 0 при переходе в состояние ESTABLISHED. Update ID действует в одной ассоциации HIP, не переходя в другие ассоциации или хосты. Update ID увеличивается на 1 перед каждым пакетом UPDATE, передаваемым хостом. В первом UPDATE от хоста Update ID = 0.

5.2.17. ACK



Type

449

Length

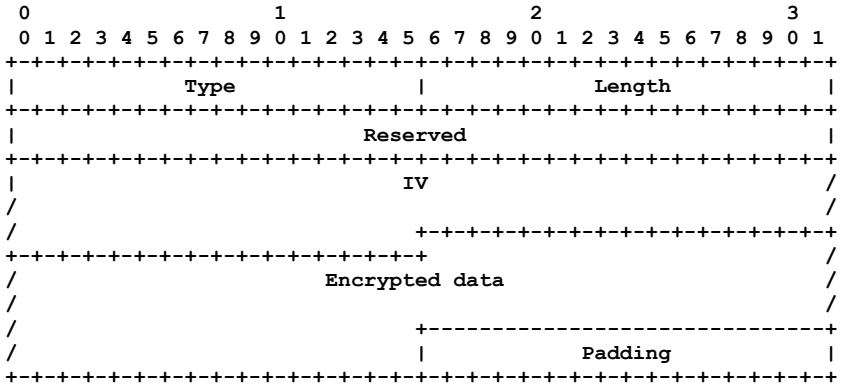
Размер в октетах без учёта Type, Length

peer Update ID

32-битовый порядковый номер, соответствующий полю Update ID в подтверждаемом пакете.

Параметр ACK включает одно или несколько полей Update ID для полученных от партнёра пакетов. Число полей определяется делением поля Length на 4.

5.2.18. ENCRYPTED



Type

641

Length

Размер в октетах без учёта Type, Length, Padding

Reserved

0 при передаче, игнорируется при получении.

IV

Вектор инициализации (при необходимости), размер которого выводится из HIP_CIPHER.

Encrypted data

Данные, зашифрованные с использованием алгоритма, указанного параметром HIP_CIPHER.

Параметр ENCRYPTED инкапсулирует другие параметры в форме зашифрованных данных (блочный шифр). Первыми полями инкапсулированных параметров являются Type и Length первого из таких параметров, что позволяет легко разобрать содержимое после расшифровки.

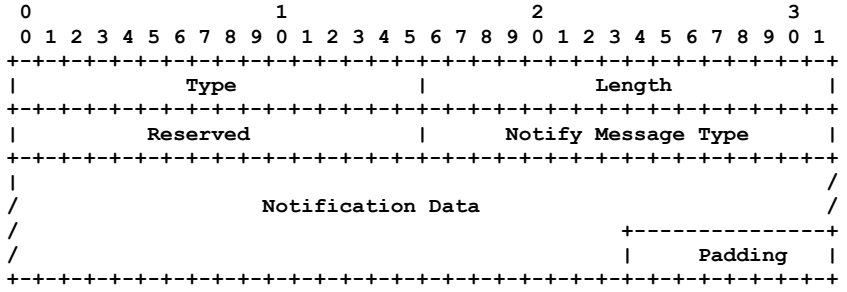
Поле Encrypted data содержит шифрованный вывод конкатенации параметров HIP. Каждый из этих «внутренних» параметров дополняется в соответствии с правилами параграфа 5.2.1. В результате размер конкатенации параметров кратен 8 байтам.

Некоторые алгоритмы шифрования требуют, чтобы размер шифруемых данных был кратным размеру блока шифрования. В этом случае упомянутый выше блок **должен** включать дополнительное заполнение до нужного размера (кратного размеру блока шифрования). Алгоритм шифрования может задавать заполнение, отличное от 0, например, в AES [FIPS.197.2001] применяется схема заполнения PKCS5 (см. параграф 6.1.1 в [RFC2898]) где для заполнения n байтов применяется значение n. Это даёт блок «нешифрованных данных», которые преобразуются в «шифрованный блок». Дополнительное заполнение добавляется к набору параметров для выполнения требований к выравниванию шифруемых блоков и не учитывается в полях HIP TLV Length, а после расшифровки удаляется.

Отметим, что размер данных после шифрования может отличаться от исходного, поскольку при шифровании может применяться сжатие или дополнение данных. По завершении процесса шифрования поле Encrypted data готово для включения в параметр. При необходимости добавляется поле Padding для выравнивания (см. параграф 5.2.1).

5.2.19. NOTIFICATION

Параметр NOTIFICATION служит для передачи партренру HIP таких сведений, как сообщения об ошибках и смене состояний. Паараметр может передаваться в пакетах NOTIFY, а возможность его применения в других типах пакетов требует изучения.



Type

832

Length

Размер в октетах без учёта Type, Length, Padding.

Reserved

0 при передаче, игнорируется при получении.

Notify Message Type

Указывает тип уведомления.

Notification Data

Информация или сведения об ошибке, дополняющие Notify Message Type и зависящие от типа (см. ниже).

Сведения из уведомлений могут включать сообщения об ошибках, указывающие причину отказа при создании HIP Security Association, а также информацию о состоянии, которую реализация HIP хочет передать партнеру. Список уведомлений со значениями Notify Message Type приведён ниже. Пакет HIP **может** включать несколько параметров NOTIFICATION при необходимости передать несколько разных уведомлений. Для предотвращения некоторых типов атак Ответчику **следует** избегать отправик NOTIFICATION хосту, который не прошёл проверку с помощью головоломки (puzzle solution).

Notify Message Type из диапазона 0-16383 предназначены для информирования об ошибках, а 16384-65535 - для сведений о состоянии. Реализации, получившей пакет NOTIFY с Notify Message Type, указывающим ошибку, в ответ на пакет запроса (например, I1, I2, UPDATE) **следует** считать, что соответствующий запрос привел к полному отказу. Неизвестные типы ошибок **должны** игнорироваться, но это **следует** вносить в системный журнал.

В настоящее время Notify Message Type из диапазона 1-10 служат для информирования об ошибках в структуре пакетов, 11-20 - о проблемах в параметрах.

Поле Notification Data в параметре NOTIFICATION, где Notify Message Type относится к диапазону состояний, **должно** игнорироваться, если оно не распознано.

Notify Message Type для ошибок

UNSUPPORTED_CRITICAL_PARAMETER_TYPE 1

Передаётся для нераспознанного параметра с установленным битом C. В поле Notification Data указывается 2-октетный типа параметра.

INVALID_SYNTAX 7

Указывает недействительность полученного сообщения HIP по причине недопустимого типа, размера или значения, а также при иных нарушениях формата. Для предотвращения DoS-атак с применением обманных сообщений этот код можно возвращать лишь для пакетов, где поля HIP_MAC (при наличии) и SIGNATURE были проверены. Этот код **должен** передаваться в отклике на любую ошибку, для которой нет отдельного кода и в него **не следует** включать детали для предотвращения утечки информации к зондирующим узлам. Для отладки **следует** записывать более подробные сведения в системный журнал.

NO_DH_PROPOSAL_CHOSEN 14

Ни один из предложенных Group ID не приемлем.

INVALID_DH_CHOSEN 15

Поле DH Group ID не соответствует предложенным Ответчиком.

NO_HIP_PROPOSAL_CHOSEN 16

Ни один из предложенных HIT Suite или HIP Encryption Algorithm не приемлем.

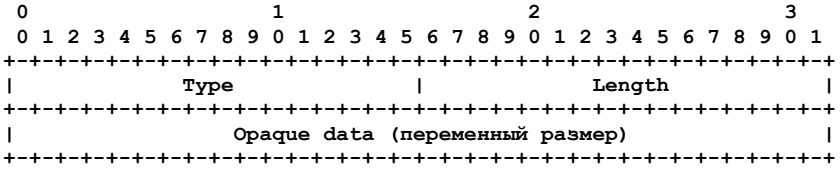
INVALID_HIP_CIPHER_CHOSEN 17

HIP_CIPHER Crypto ID не соответствует предложенным Ответчиком.

UNSUPPORTED_HIT_SUITE 20
Передаётся в ответ на I1 или R1, для которого HIT Suite не поддерживается.
AUTHENTICATION_FAILED 24
Передаётся при ошибке в подписи HIP, за исключением случаев отказа при проверке подписи в NOTIFY.
CHECKSUM_FAILED 26
Передаётся при ошибке в контрольной сумме HIP.
HIP_MAC_FAILED 28
Передаётся при ошибке в HIP HMAC.
ENCRYPTION_FAILED 32
Ответчик не смог расшифровать параметр ENCRYPTED.
INVALID_HIT 40
Передаётся при отказе в процессе проверки HIT партнёра по соответствующему HI.
BLOCKED_BY_POLICY 42
Ответчик не хочет воспринимать ассоциацию в соответствии с политикой (например, при получении HIT = NULL, если политика не разрешает режим Opportunistic).
RESPONDER_BUSY_PLEASE_RETRY 44
Ответчик не хочет воспринимать ассоциацию по причине перегрузки и желания снизить загрузку, отклонив запрос Инициатора. Инициатор может повторить попытку, однако он **должен** найти другое (отличающееся) решение гловоломки для такого повтора. Отметим, что Инициатору может потребоваться получение новой головоломки с помощью обмена I1/R1.
Notify Message Type для состояния

I2_ACKNOWLEDGEMENT 16384
Ответчик получил пакет I2 от Инициатора, но его пришлось поместить в очередь на обработку. Головоломка была решена верно и Ответчик хочте организовать ассоциацию, но в данный момент в очереди находится некотрое число пакетов I2. Пакет R2 передаётся после обработки I2.

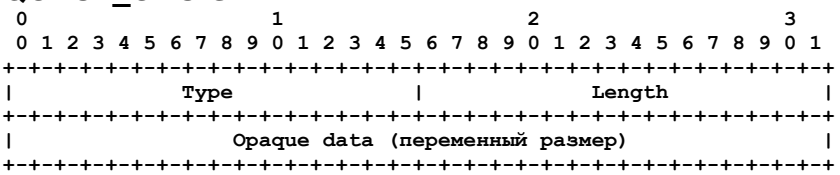
5.2.20. ECHO_REQUEST_SIGNED



Type
897
Length
Размер Opaque data в октетах.
Opaque data
Неанализируемый данные, которые имеют смысл лишь для узла, передавшего ECHO_REQUEST_SIGNED и получившего в ответ ECHO_RESPONSE_SIGNED или ECHO_RESPONSE_UNSIGNED
Параметр ECHO_REQUEST_SIGNED содержит неанализируемый блок данных (blob) которые отправитель хочет получить обраьно в соответствующем отклике.

ECHO_REQUEST_SIGNED и соответствующие параметры отклика **могут** использоваться узлом для любых целей, когда он хочет передать то или иное состояние в запросе и получить его обратно в пакете отклика. Параметр ECHO_REQUEST_SIGNED учитывается в HIP_MAC и SIGNATURE. Пакет HIP может содержать лишь 1 параметр ECHO_REQUEST_SIGNED и **может** включать несколько параметров ECHO_REQUEST_UNSIGNED. Откликом на ECHO_REQUEST_SIGNED **должен** быть ECHO_RESPONSE_SIGNED.

5.2.21. ECHO_REQUEST_UNSIGNED

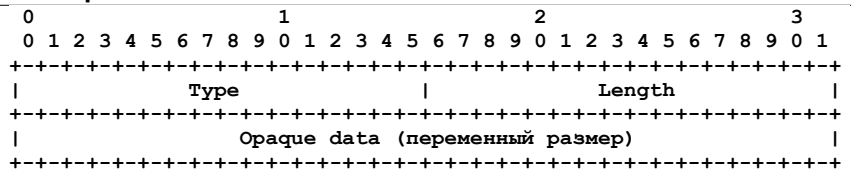


Type
63661
Length
Размер Opaque data в октетах.
Opaque data
Неанализируемые (opaque) данные, которые предполагаются осмысленными лишь для узла, передавшего ECHO_REQUEST_UNSIGNED и получающего соответствующий параметр ECHO_RESPONSE_UNSIGNED.
ECHO_REQUEST_UNSIGNED содержит неанализируемый блок данных (blob), которые отправитель желает получить обратно в отклике.

ECHO_REQUEST_UNSIGNED и соответствующий отклик **могут** служить для любых целей, когда узел хочет передать в запросе некоторое состояние и получить его обратно в пакете отклика. ECHO_REQUEST_UNSIGNED не учитывается в HIP_MAC и SIGNATURE. Пакет HIP может включать 1 или несколько параметров ECHO_REQUEST_UNSIGNED. Промежуточные устройства могут добавлять параметры ECHO_REQUEST_UNSIGNED в проходящие через них пакеты HIP. Создатель ECHO_REQUEST_UNSIGNED (конечный хост или промежуточное устройство) должен создавать поле Opaque так, чтобы можно было обнаружить и удалить соответствующий параметр ECHO_RESPONSE_UNSIGNED.

В ответ на параметр ECHO_REQUEST_UNSIGNED **должен** передаваться параметр ECHO_RESPONSE_UNSIGNED.

5.2.22. ECHO_RESPONSE_SIGNED

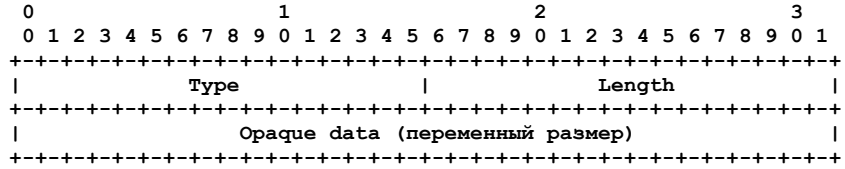


Type
961
Length
Размер Opaque data в октетах.
Opaque data
Неанализируемые (opaque) данные, которые копируются из параметра ECHO_REQUEST_SIGNED или ECHO_REQUEST_UNSIGNED, вызвавшего этот отклик.

ECHO_RESPONSE_SIGNED содержит неанализируемый блок данных (blob), которые отправитель ECHO_REQUEST_SIGNED желает получить обратно. Эти данные копируются из параметра ECHO_REQUEST_SIGNED.

Параметры ECHO_REQUEST_SIGNED и ECHO_RESPONSE_SIGNED **могут** служить для любых целей, когда узел хочет передать в запросе некоторое состояние и получить его обратно в пакете отклика. ECHO_RESPONSE_SIGNED учитывается в HIP_MAC и SIGNATURE.

5.2.23. ECHO_RESPONSE_UNSIGNED



Type
63425
Length
Размер Opaque data в октетах.
Opaque data
Неанализируемые (opaque) данные, которые копируются из параметра ECHO_REQUEST_SIGNED или ECHO_REQUEST_UNSIGNED, вызвавшего этот отклик.

ECHO_RESPONSE_UNSIGNED содержит неанализируемый блок данных (blob), которые отправитель ECHO_REQUEST_SIGNED или ECHO_REQUEST_UNSIGNED желает получить обратно. Эти данные копируются из соответствующего параметра запроса.

Запрос и ECHO_RESPONSE_UNSIGNED **могут** служить для любых целей, когда узел хочет передать в запросе некоторое состояние и получить его обратно в пакете отклика. ECHO_RESPONSE_UNSIGNED не учитывается в HIP_MAC и SIGNATURE.

5.3. Пакеты HIP

Имеется 8 основных пакетов HIP (Таблица 11), из которых 4 обеспечивают базовый обмен HIP, 1 служит для обновления, 1 - для уведомлений и 2 для закрытия ассоциации HIP. Поддержка пакетов типа NOTIFY не обязательна, остальные пакеты HIP, перечисленные в таблице 11, должны поддерживаться.

Таблица 11. Типы пакетов HIP.

Тип пакета	***Название пакета***
1	I1 - пакет Инициатора HIP
2	R1 - пакет Ответчика HIP
3	I2 - второй пакет Инициатора HIP
4	R2 - второй пакет Ответчика HIP
16	UPDATE - пакет HIP Update
17	NOTIFY - пакет HIP Notify
18	CLOSE - пакет HIP Association Closing
19	CLOSE_ACK - пакет HIP Closing Acknowledgment

Пакеты включают фиксированный заголовок (параграф 5.1), за которым могут следовать параметры в формате TLV.

В дополнение к базовым пакетам дополнительные спецификации могут определять другие типы пакетов. Например, поддержка многоадресности и мобильности не включена в эту спецификацию.

Используемые в описаниях пакетов обозначения приведены в параграфе 2.2.

В будущем после заголовка HIP **могут** быть размещены данные вышележащего уровня. Поле Next Header в заголовке указывает наличие дополнительных данных после заголовка HIP. Пакеты HIP **недопустимо** фрагментировать на несколько заголовков расширения путём установки в поле Next Header заголовка HIP номера протокола HIP. Это ограничивает размер дополнительных данных в пакете.

5.3.1. I1 - пакет Инициатора HIP

Заголовок HIP для пакета I1 имеет вид

```
Packet Type = 1
SRC HIT = HIT Инициатора
DST HIT = HIT Ответчика или NULL

IP ( HIP ( DH_GROUP_LIST ) )
```

Пакет I1 содержит фиксированный заголовок HIP и список Инициатора DH_GROUP_LIST. Битов управления нет.

Инициатор получает HIT Ответчика из запроса DNS или FQDN Ответчика (см. [HIP-DNS-EXT]), иного репозитория или локальной таблицы. Если Инициатор не знает HIT Ответчика, от может проробовать режим Opportunistic (параграф 4.1.8), установив значение NULL (все нули) как HIT Ответчика.

Поскольку пакет I1 легко подделать (даже подписанный), не предпинимается попыток усложнить его создание и обработку.

Инициатор включает параметр DH_GROUP_LIST в пакет I1 для информирования ответчика о предпочитаемых DH Group ID. Отметим, что DH_GROUP_LIST в пакетах I1 не учитывается в подписи пакета.

Реализации **должны** быть способны обрабатывать «шквал» пакетов I1, отбрасывая одинаковые пакеты, прибывшие близко по времени.

5.3.2. R1 - пакет Ответчика HIP

Заголовок HIP для пакета R1 имеет вид

```
  Packet Type = 2
  SRC HIT = HIT Ответчика
  DST HIT = HIT Инициатора

IP ( HIP ( [ R1_COUNTER, ]
           PUZZLE,
           DIFFIE_HELLMAN,
           HIP_CIPHER,
           HOST_ID,
           HIT_SUITE_LIST,
           DH_GROUP_LIST,
           [ ECHO_REQUEST_SIGNED, ]
           TRANSPORT_FORMAT_LIST,
           HIP_SIGNATURE_2 )
           <, ECHO_REQUEST_UNSIGNED >i)
```

Бит управления A **должен** устанавливаться если идентификатор HI Ответчика является анонимным.

Тег HIT Инициатора **должен** совпадать с одним из полученных в пакете I1, если R1 является откликом на I1. Если у Ответчика имеется несколько HI, используемый HIT Ответчика **должен** соответствовать запросу Инициатора. Если Инициатор использует режим Opportunistic, Ответчик может свободно выбирать из HI (см. параграф 4.1.8).

Счётчик генерации пакетов R1 служит для определения текущего действительного поколения головоломок. Значение счётчика увеличивается периодически и **рекомендуется** увеличивать его не реже, чем перестают восприниматься решения прежних голововломок. Головоломка содержит случайное значение Random #I и сложность #K. Сложность указывает число младших битов в результате хэширования головоломки, которые должны быть нулями (см. параграф 4.1.2). Random #I не учитывается в подписи и должно считаться 0 при создании подписи, чтобы отправитель мог установить #I в подготовленном заранее пакете R1 перед его отправкой партнеру.

Ответчик выбирает DIFFIE_HELLMAN Group ID и Public Value в соответствии с предпочтениями Инициатора, указанными параметром DH_GROUP_LIST в пакете I1. Ответчик возвращает свои предпочтения, на основе которых он выбрал открытое значение DH, в DH_GROUP_LIST. Это позволяет Инициатору определить, не был ли список DH_GROUP_LIST в его пакете I1 изменён атакующим.

Открытое значение DH является эфемерным и значения **не следует** неоднократно применять в разных ассоциациях HIP. Как только Ответчик получил действительный отклик на пакет R1, значение DH **следует** исключить. Возможна передача Ответчиком одного значения DH одновременно разным хостам в соответствующих пакетах R1 и такие отклики также следует воспринимать. Однако в качестве защиты от «шквала» пакетов I1 реализация **может** предложить и неоднократно использовать одно значение DH (если этого не избежать) в течение некоторого времени, например, 15 минут. Используя небольшое число разных головоломок для данного значения DH, пакеты R1 можно подготовить заранее и передавать со скоростью получения пакетов I1. Процессу сборки мусора (scavenger) следует очищать неиспользуемые значения DH и головоломки.

Повторное использование открытых значений DH создаёт риск безопасности, связанный с использованием несколькими Инициаторами одного ключевого материала (из-за несовершенства генераторов случайных чисел). Кроме того, использование несколькими Инициаторами одной половины открытого ключа Ответчика может приводить к упрощения криптографических атак и несовершенству защиты.

Однако риски, связанные с повторным использованием одного значения, носят статистический характер и авторам неизвестен какой-либо механизм, позволяющий манипулировать протоколом так, чтобы риск повторного использования любого данного ключа DH Ответчика отличался от базовой вероятности. Поэтому Ответчикам **рекомендуется** избегать повторного применения одного ключа DH для нескольких Инициаторов, но (поскольку риск считается статистическим и возможности манипуляций неизвестны), реализации **могут** повторно использовать ключ для экономии ресурсов и повышения вероятности успешного взаимодействия с традиционными клиентами даже при шквале пакетов I1. В частности, когда слишком расточительно заранее создавать пакеты R1 для каждого возможного Инициатора со своим ключом DH, Ответчик **может** передать один ключ DH нескольким Инициаторам, создавая возможность применения несколькими легитимными Инициаторами одного ключа на стороне ответчика. Однако, как только Ответчик узнает об использовании конкретного ключа DH, ему **следует** прекратить предложение этого ключа. Это направлено на обеспечение Ответчикам с ограниченными ресурсами возможности предлагать услуги при шквалах пакетов I1 и одновременно сделать возможность повторного использования ключа DH статистической и как можно более низкой.

Если Ответчик применяет одну пару ключей DH для нескольких согласования, он должен позаботиться о предотвращении атак на мелкие подгруппы [RFC2785]. Для этого при получении I2 Ответчику **следует** проверить открытый ключ DH Инициатора, как указано в параграфе 3.1 [RFC2785]. Если проверка не прошла, Ответчику **недопустимо** создавать общий ключ DH и он должен прервать HIP BEX.

Параметр HIP_CIPHER содержит алгоритмы, поддерживаемые Ответчиком для шифрования содержимого параметра ENCRYPTED, в порядке предпочтительности. Все реализации **должны** поддерживать AES [RFC3602].

Параметр HIT_SUITE_LIST содержит упорядоченный список предпочитаемых и поддерживамых Ответчиком HIT Suite. Этот список позволяет Инициатору определить соответствует ли его HIT источника поддерживаемому Ответчиком набору.

Параметры ECHO_REQUEST_SIGNED и ECHO_REQUEST_UNSIGNED содержат данные, которые отправитель хочет получить неизменными обратно в параметре ECHO_RESPONSE_SIGNED или ECHO_RESPONSE_UNSIGNED. Пакет R1 можен (но не обязан) влючать один или несколько параметров ECHO_REQUEST_UNSIGNED, как указано в параграфе 5.2.21.

Параметр TRANSPORT_FORMAT_LIST содержит упорядоченный список предпочитаемых и поддерживаемых Ответчиком транспортных форматов. Этот список позволяет Инициатору и Ответчику согласовать общий способ защиты данных (payload). Параметр описан в параграфе 5.2.11.

Подпись рассчитывается для пакета HIP, как указано в параграфе 5.2.15. Это Ответчику заранее создать пакеты R1. Инициатору **следует** проверять подпись и он **должен** проверить соответствие HI Ответчика ожидаемому значению.

5.3.3. I2 - второй пакет Инициатора HIP

Заголовок HIP для пакета I2 имеет вид

```
      Packet Type = 3
      SRC HIT = HIT Инициатора
      DST HIT = HIT Ответчика

    IP ( HIP ( [R1_COUNTER,]
               SOLUTION,
               DIFFIE_HELLMAN,
               HIP_CIPHER,
               ENCRYPTED { HOST_ID } or HOST_ID,
               [ ECHO_RESPONSE_SIGNED, ]
               TRANSPORT_FORMAT_LIST,
               HIP_MAC,
               HIP_SIGNATURE
               <, ECHO_RESPONSE_UNSIGNED>i ) )
```

Действителен бит управления A, который **должен** устанавливаться, если HI Инициатора является анонимным.

Теги HIT **должны** соответствовать указанным в R1.

При наличии в пакете R1 параметра R1_COUNTER Инициатор **должен** скопировать полученный в R1 параметр в I2.

Параметр Solution содержит Random #I из пакета R1 и рассчитанное значение #J. Младшие #K битов RHASH(#I | ... | #J) **должны** иметь значение 0.

Значение DH является эфемерным. Если расчёт выполняется заранее, процессу сборки мусора следует очищать неиспользуемые значения DH. Ответчик **может** повторно использовать значения DH при некоторых условиях, как указано в параграфе 5.3.2.

HIP_CIPHER содержит один шифронабор, выбранный Инициатором, который тот использует для шифрования параметров ENCRYPTED. Выбранный шифр **должен** соответствовать одному из предложенных Ответчиком в пакете R1. Все реализации **должны** поддерживать AES [RFC3602].

HI Инициатора **может** быть зашифрован с использованием алгоритма HIP_CIPHER. Ключевой материал выводится из обмена DH, как указано в параграфе 6.5.

ECHO_RESPONSE_SIGNED и ECHO_RESPONSE_UNSIGNED содержат неанализируемые (opaque) данные из соответствующих параметров эхо-запроса.

TRANSPORT_FORMAT_LIST содержит один тип формата транспорта, выбранный Инициатором. Выбранный тип **должен** соответствовать одному из предложенных Ответчиком в пакете R1. В настоящее премя определён лишь транспортный формат ESP ([RFC7402]).

Значение HMAC в параметре HIP_MAC рассчитывается для всего пакета HIP за исключением параметров после HIP_MAC, как указано в параграфе 6.4.1. Ответчик **должен** проверять HIP_MAC.

Подпись рассчитывается для всего пакета HIP за исключением параметров после HIP_SIGNATURE, как описано в параграфе 5.2.14. Ответчик **должен** проверять подпись, используя HI из пакета или идентификатор HI полученный иным способом.

5.3.4. R2 - второй пакет Ответчика HIP

Заголовок HIP для пакета R2 имеет вид

```
      Packet Type = 4
      SRC HIT = HIT Ответчика
      DST HIT = HIT Инициатора

    IP ( HIP ( HIP_MAC_2, HIP_SIGNATURE ) )
```

Битов управления нет.

HIP_MAC_2 рассчитывается для всего пакета HIP объединённого (конкатенация) с параметром Ответчика HOST_ID. После расчёта HMAC параметр HOST_ID удаляется, процедура описана в параграфе 6.4.1.

Подпись рассчитывается для всего пакета HIP.

Инициатор **должен** проверять HIP_MAC и подпись.

5.3.5. UPDATE - пакет обновления HIP

Заголовок HIP для пакета UPDATE имеет вид

```
      Packet Type = 16
      SRC HIT = HIT отправителя
      DST HIT = HIT получателя

    IP ( HIP ( [SEQ, ACK, ] HIP_MAC, HIP_SIGNATURE ) )
```

Битов управления нет.

Пакет UPDATE содержит обязательные параметры HIP_MAC и HIP_SIGNATURE, а также необязательные параметры.

Пакет UPDATE может включать 1 параметр SEQ, наличие которого указывает, что получатель **должен** подтвердить UPDATE. Пакет UPDATE без параметра SEQ, содержащий лишь параметр ACK является просто подтверждением предыдущего UPDATE и его **недопустимо** подтверждать отдельным параметром ACK. Пакеты UPDATE, содержащие лишь параметр ACK, не требуют обработки в порядке, соответствующем другим пакетам UPDATE. Пакет UPDATE, не включающий параметр SEQ или ACK, является недействительным и для таких неподтверждаемых обновлений **должны** применяться пакеты NOTIFY.

Пакет UPDATE может включать параметр ACK, содержащий порядковый номер SEQ из подтверждаемого пакета UPDATE. Хост **может** подтверждать сразу несколько пакетов UPDATE, например, параметр ACK может содержать два полученных последними значения SEQ для устойчивости к потере пакетов. Значения ACK не кумулятивны и для каждого полученного уникального SEQ требуется по меньшей мере одно соответствующее значения ACK в отклике. Полученные избыточные параметры ACK игнорируются. Хосты **должны** реализовать обработку параметров ACK с несколькими порядковыми номерами SEQ, даже если они не реализуют отправку параметров ACK с несколькими SEQ.

Пакет UPDATE может включать оба параметра SEQ и ACK. В общем случае пакеты UPDATE с параметром SEQ **следует** подтверждать по завершении обработки UPDATE. Хост **может** задержать UPDATE с параметром ACK на короткое время, чтобы совместить ACK, подобно отложенным подтверждениям в TCP.

Отправитель **может** отказаться от гарантированной передачи конкретного UPDATE (например, при перегрузке событиями). В соответствии с семантикой получатель **должен** подтверждать UPDATE, но отправитель **может** не заботиться о получении параметра ACK.

Пакеты UPDATE **можно** передавать повторно без увеличения SEQ. Если одно подмножество параметров включено в несколько пакетов UPDATE с разными SEQ, хост **должен** гарантировать, что многократная обработка параметров не приведёт к протокольной ошибке.

5.3.6. NOTIFY - пакет уведомления HIP

Пакет NOTIFY **может** служить для предоставления информации партнёру. Обычно пакеты NOTIFY применяются для указания некоторых ошибок протокола или отказов при согласовании. Пакеты NOTIFY не подтверждаются. Получатель может обработать пакет лишь как информационный и **не следует** менять состояние HIP (см. параграф 4.4.2) на основе лишь получения пакета NOTIFY.

Заголовок HIP для пакета NOTIFY имеет вид

```
      Packet Type = 17
      SRC HIT = HIT отправителя
      DST HIT = HIT получателя или 0, если тег неизвестен

    IP ( HIP (<NOTIFICATION>i, [HOST_ID, ] HIP_SIGNATURE) )
```

Битов управления нет.

Пакеты NOTIFY служат для передачи одного или нескольких параметров NOTIFICATION.

5.3.7. CLOSE - пакет закрытия ассоциации HIP

Заголовок HIP для пакета CLOSE имеет вид

```
      Packet Type = 18
      SRC HIT = HIT отправителя
      DST HIT = HIT получателя

    IP ( HIP ( ECHO_REQUEST_SIGNED, HIP_MAC, HIP_SIGNATURE ) )
```

Битов управления нет.

Отправитель **должен** включать параметр ECHO_REQUEST_SIGNED, используемый для проверки CLOSE_ACK, полученного в отклие, а также HIP_MAC и подпись (для всего пакета HIP).

Получатель **должен** отвечать пакетом CLOSE_ACK с параметром ECHO_RESPONSE_SIGNED, соответствующим ECHO_REQUEST_SIGNED.

5.3.8. CLOSE_ACK - пакет подтверждения закрытия HIP

Заголовок HIP для пакета CLOSE_ACK имеет вид

```
      Packet Type = 19
      SRC HIT = HIT отправителя
      DST HIT = HIT получателя

    IP ( HIP ( ECHO_RESPONSE_SIGNED, HIP_MAC, HIP_SIGNATURE ) )
```

Битов управления нет.

Отправитель **должен** включать в пакет HMAC и подпись (для всего пакета HIP). Получатель **должен** проверить ECHO_RESPONSE_SIGNED, а также HIP_MAC и подпись, если у него есть статус ассоциации HIP.

5.4. Сообщения ICMP

Когда реализация HIP обнаруживает проблему со входящим пакетом и не может определить отправителя или не имееет с отправителем ассоциации HIP, она **может** ответить пакетом ICMP. Частота отправки таких откликов **должна** быть ограничена в соответствии с [RFC4443]. В большинстве случаев пакет ICMP имеет тип Parameter Problem (12 для ICMPv4, 4 для ICMPv6) с полем Pointer, указывающим вызвавшее генерацию сообщения ICMP поля.

5.4.1. Недействительная версия

Если реализация HIP получает пакет HIP с нераспознанным номером версии HIP, ей **следует** отвечать пакетом ICMP типа Parameter Problem с полем Pointer, указывающим байт Version/RES. в заголовке пакета HIP. Частоту передачи таких пакетов **следует** ограничивать.

5.4.2. Другие проблемы в заголовке HIP и структуре пакета

Реализация HIP, получившая пакет HIP с нерешаемыми проблемами в заголовке или формате, она **может** ответить пакетом ICMP типа Parameter Problem с полем Pointer, указывающим проблемное место в пакете. Частоту передачи таких пакетов следует ограничивать. Реализации **недопустимо** передавать сообщения ICMP при ошибке контрольной суммы, такие пакеты **должны** просто отбрасываться.

5.4.3. Неверное решение Puzzle

Если реализация HIP получает пакет I2 с неверным решением головоломки, её поведение зависит от версии протокола IP. В случае IPv6 реализации **следует** отвечать пакетом ICMP типа Parameter Problem с полем Pointer, указывающим начало поля Puzzle solution #J в параметре SOLUTION пакета HIP. При использовании IPv4 реализация **может** ответить пакетом ICMP типа Parameter Problem, копируя из пакета I2 достаточно байтов, чтобы поместить параметр SOLUTION в сообщение ICMP и указывая в поле Pointer начало поля Puzzle solution #J, как для IPv6. Отметим, что размер сообщения ICMPv4 будет превышать обычный размер ICMPv4, определённый в [RFC0792].

5.4.4. Несуществующая ассоциация HIP

Если реализация HIP получает пакет CLOSE, UPDATE или иной пакет, для обслуживания которого нужна ассоциация, а тег HIT отправителя или получателя не соответствует имеющимся ассоциациям HIP, реализация **может** ответить пакетом ICMP типа Parameter Problem, ограничивая скорость передачи таких пакетов. Поле Pointer в пакете ICMP Parameter Problem должно указывать начало первого тега HIT, который не соответствует имеющимся ассоциациям.

Хосту **недопустимо** отвечать такими сообщениями ICMP на пакеты I1, R2, I2, R2, NOTIFY. При определении нового типа пакетов в спецификации **следует** указать правила в части передачи этого типа сообщений ICMP.

6. Обработка пакетов

На каждом хосте предполагается одна реализация HIP, которая поддерживает HIP-ассоциации хоста и обслуживает запросы на создание новых ассоциаций. Каждая ассоциация управляется концептуальным конечным автоматом, состояния которого определены в параграфе 4.4. Реализация HIP может одновременно поддерживать ассоциации HIP с несколькими хостами и даже может иметь несколько активных ассоциаций с одним хостом, различаемых по тегам HIT. Однако для любой пары HIT невозможно создать более одной ассоциации HIP одновременно, поэтому единственным способом организации между парой хостов нескольких ассоциаций является использование разных HIT.

Обработка пакетов зависит от состояния ассоциаций HIP по отношению к аутентифицированному или представляющемуся инициатором партнёру. Реализация HIP проверяет наличие активной ассоциации с отправителем пакета на основе тегов HIT. Если данные передаются в конкретном транспортном формате, сопоставление пакетов с имеющимися ассоциациями задаёт спецификация транспортного формата.

6.1. Обработка исходящих данных приложения

Приложение на хосте HIP может передавать свои данные, используя идентификатор, заданный через базовый API. Интерфейс API может быть совместимым с прежними версиями (см. [RFC5338]), используя идентификаторы подобно адресам IP, или полностью новым, обеспечивающим расширенные функции для Host Identity. В зависимости от реализации HIP предоставляемый приложению идентификатор может различаться, например, это может быть HIT или адрес IP.

Точный формат и метод передачи пользовательских данных от хоста-источника HIP к адресату HIP определяется соответствующей спецификацией транспортного формата. В сеть данные передаются с использованием подходящих IP-адресов отправителя и получателя.

В этом документе заданы концептуальные правила обработки для базового случая, где оба хоста имеют лишь по одному применимому адресу IP, а многоадресные многодомные системы рассматриваются в отдельном документе.

Приведённый ниже концептуальный алгоритм задаёт этапы обработки исходящих дейтаграмм, направленных по HIT.

1. Если в дейтаграмме задан конкретный адрес источника, это **должен** быть тег HIT. В ином случае реализация **может** заменить адрес отправителя тегом HIT. Иначе пакет **должен** отбрасываться.
2. Если в дейтаграмме адрес источника не задан, реализация **должна** выбрать для дейтаграммы подходящий тег HIT в соответствии с локальной политикой.
3. Если для данной пары тегов HIT отправителя и получателя нет активной ассоциации HIP, она **должна** создаваться путём запуска базового обмена. В процессе обмена реализации **следует** помещать в очередь на отправку хотя бы один пользовательский пакет для каждой создаваемой ассоциации HIP и **можно** помещать в очередь большее число пакетов.
4. При наличии активной ассоциации HIP для данной пары HIT отправителя и получателя исходящая дейтаграмма передаётся на транспортную обработку. Форматы транспорта определяют отдельные документы, а транспорт ESP для HIP является обязательным во всех реализациях HIP.

5. Перед отправкой пакета теги HIT в дейтаграмме заменяются адресами IP. Для IPv6 **следует** применять правила [RFC6724]. Отметим, что этап преобразования HIT в адрес IP **может** выполняться в другой точке стека протоколов, например, перед инкапсуляцией пакета в выходной формат.

6.2. Обработка входящих данных приложения

Приведённый ниже концептуальный алгоритм задаёт этапы обработки входящих дейтаграмм при использовании на принимающем хосте тегов HIT в качестве идентификаторов на уровне приложения. Более подробные описания обработки пакетов задают спецификации соответствующих транспортных форматов.

1. Входящая дейтаграмма сопоставляется с имеющимися ассоциациями HIP обычно с использованием информации из пакета. Например, это можно сделать по ESP SPI (Security Parameter Index).
2. Транспортный формат декапсулируется соответствующим способом, давая в результате пакет, подобный обычному (нешифрованному) пакету IP. По возможности на этом этапе **следует** также проверить, был ли пакет действительно передан (однократно) удаленным хостом HIP, указанным ассоциацией HIP. Метод проверки может меняться в зависимости от режима транспорта. Хотя HI (а также HIT) применяется в качестве идентификатора вышележащего уровня, метод проверки должен подтвердить, что пакет был передан узлом с корректным отождествлением, которое действительно отобразается на соответствующий тег HIT. Для транспорта ESP [RFC7402] проверка выполняется с использованием значения SPI в пакете данных путём поиска соответствующей защищённой связи SA со связанным тегом HIT и ключом, а также расшифровкой пакета с использованием связанного ключа.
3. Адрес IP в дейтаграмме заменяется тегом HIT, связанным с ассоциацией HIP. Отметим, что преобразование адреса IP в тег HIT **может** выполняться в другом месте стека протоколов.
4. Дейтаграмма доставляется вышележащему уровню (например, UDP или TCP). При демультиплексировании дейтаграммы сокет вылележащего уровня выбирается по тегу HIT.

6.3. Решение головоломки

В этом параграфе рассматривается решение головоломки (puzzle). В пакете R1 значения #I и #K передаются в сетевом порядке байтов, равно как значения #I и #J в пакете I2. Хэш-значение создаётся путём применения алгоритма RHASH к конкатенации (в сетевом порядке) указанных ниже данных.

n-битовое случайное значение #I (n = RHASH_len) с сетевым порядком байтов как в пакетах R1 и I2.

128-битовый тег HIT Инициатора с сетевым порядком байтов как в HIP Payload внутри пакетов R1 и I2.

128-битовый тег HIT Ответчика с сетевым порядком байтов как в HIP Payload внутри пакетов R1 и I2.

n-битовое случайное значение #J (n = RHASH_len) с сетевым порядком байтов как в пакетах R1 и I2.

В действительном решении головоломки #K младших битов результирующего дайджеста RHASH **должны** быть 0.

Примечания.

i) Размер хэшируемых данных зависит от размера результата хэш-функции RHASH у Ответчика.

ii) Все данные на входе функции хэширования **должны** использовать сетевой порядок байтов.

iii) Порядок тегов HIT Инициатора и Ответчика различается в пакетах R1 и I2 (см. параграф 5.1). Нужно соблюдать осторожность при копировании значений на вход функции хэширования.

iv) Для головоломки #I может существовать множество верных решений #J.

Ниже указаны процедуры обработки в предположении использования Ответчиком заранее созданных пакетов R1.

Предварительные расчеты Ответчика

Установка сложности головоломки #K.
Создание подписанных пакетов R1 и кэширование их.

Ответчик

Выбор подходящего R1 из кэша.
Генерация случайного значения #I.
Отправка #I и #K в пакете R1.
Сохранение #I и #K на некоторое время (delta).

Инициатор

Повторение попыток решить головоломку, пока не будет найдено соответствующее значение #J:
Ltrunc(RHASH(#I | HIT-I | HIT-R | #J), #K) == 0
Отправка #I и #J в пакете I2.

Ответчик

Проверка того, что полученное значение #I было сохранено.
Определение верного #K на основе #I.
Расчет V := Ltrunc(RHASH(#I | HIT-I | HIT-R | #J), #K)
Отключение при V != 0
Восприятие при V == 0

6.4. Расчёт и проверка HIP_MAC и SIGNATURE

В последующих параграфах определены действия по обработке параметров HIP_MAC, HIP_MAC_2, HIP_SIGNATURE, HIP_SIGNATURE_2. Параметр HIP_MAC_2 содержится в пакетах R2, HIP_SIGNATURE_2 - в R1, а HIP_SIGNATURE и HIP_MAC - в других пакетах HIP.

6.4.1. Расчёт HMAC

Для HMAC применяется функция хэширования RHASH, тип которой зависит от HIT Suite у Ответчика. В HIT Suite RSA/DSA/SHA-256 применяется HMAC-SHA-256 [RFC4868], в ECDSA_LOW/SHA-1 - HMAC-SHA-1 [RFC2404], в ECDSA/SHA-384 - HMAC-SHA-384 [RFC4868].

Описанный ниже процесс применим к параметрам HIP_MAC и HIP_MAC_2. При обработке HIP_MAC_2 отличие от расчётов для HIP_MAC включает псевдо-поле HOST_ID с информацией Ответчика, переданной ранее в пакете R1.

Инициатору и Ответчику следует проявлять осторожность при проверке и расчёте HIP_MAC_2. В частности, Инициатор должен сохранять HOST_ID в точности, как было в пакете R1, пока не получит параметр HIP_MAC_2 в пакете R2.

Расчёт HIP_MAC выполняется как

```
HMAC: { HIP header | [ Parameters ] }
```

где Parameters включает все параметры пакета HIP с типами от 1 до (тип HIP_MAC - 1).

При расчёте HIP_MAC применяются два правила:

- в заголовке HIP устанавливается Checksum = 0;
- в заголовке HIP отсчёт поля Header Length начинается от начала параметра HIP_MAC.

Порядок параметров описан в параграфе 5.2.1.

Расчёт HIP_MAC_2 выполняется как

```
HIP_MAC_2: { HIP header | [ Parameters ] | HOST_ID }
```

где Parameters включает все параметры пакета HIP с типами от 1 до (тип HIP_MAC_2 - 1).

При расчёте HIP_MAC_2 применяются три правила:

- в заголовке HIP устанавливается Checksum = 0;
- в заголовке HIP отсчёт поля Header Length начинается от начала параметра HIP_MAC_2 и увеличивается на размер конкатенации HOST_ID (включая поля Type и Length);
- параметр HOST_ID имет именно ту форму, в которой он был получен в пакете R1 от Ответчика.

Порядок параметров описан в параграфе 5.2.1, за исключением того, что параметр HOST_ID в этом расчёте добавляется в конце.

Параметр HIP_MAC определён в параграфе 5.2.12, а HIP_MAC_2 - в параграфе 5.2.13. Процесс расчёта и проверки HMAC (для HIP_MAC и HIP_MAC_2, пока для HIP_MAC_2 не указано иное) описан ниже

Отправитель пакета

1. Создаётся пакет HIP без HIP_MAC, HIP_SIGNATURE, HIP_SIGNATURE_2 и иных параметров, значение типа которых превышает значение типа параметра HIP_MAC.
2. При расчёте HIP_MAC_2 в конец пакета добавляется параметр HOST_ID (Ответчик).
3. При расчёте HIP_MAC_2 в поле Header Length заголовка HIP учитывается параметр HOST_ID.
4. Рассчитывается HMAC с использованием ключа HIP-gl или HIP-lg, выведенного из KEYMAT в соответствии с параграфом 6.5.
5. Для HIP_MAC_2 параметр HOST_ID удаляется из пакета.
6. В пакет добавляется параметр HIP_MAC и другие параметры, значение типа которых больше значения типа HIP_MAC (HIP_MAC_2), включая возможные параметры HIP_SIGNATURE и HIP_SIGNATURE_2.
7. Заново рассчитывается значение поля Length в заголовке HIP.

Получатель пакета

1. Проверка поля HIP Header Length.
2. Удаляется параметр HIP_MAC или HIP_MAC_2, а также последующие параметры с большими значениями типа, включая возможные параметры HIP_SIGNATURE или HIP_SIGNATURE_2 с сохранением содержимого, которое может потребоваться позднее.
3. В случае HIP_MAC_2 создаётся и добавляется в пакет параметр HOST_ID (с информацией Ответчика). Параметру HOST_ID следует совпадать с полученные ранее от Ответчика значением.
4. Пересчитывается размер пакета HIP в заголовке HIP и очищается поле Checksum (0). В случае HIP_MAC_2 размер рассчитывается с учетом поля HOST_ID.
5. Рассчитывается HMAC с использованием ключа HIP-gl или HIP-lg, выведенного из KEYMAT в соответствии с параграфом 6.5 и сравнивается с полученным HMAC.
6. В полях Checksum и Header Length заголовка HIP устанавливаются исходные значения. Отметим, что значения Checksum и Length после этого становятся некорректными.
7. В случае HIP_MAC_2 из пакета удаляется параметр HOST_ID перед дальнейшей обработкой.

6.4.2. Расчёт подписи

Приведённая ниже процедура применима к параметрам HIP_SIGNATURE и HIP_SIGNATURE_2. При обработке HIP_SIGNATURE_2 отличием является использование параметра HIP_SIGNATURE_2 и HIT Инициатора, а также очистка (0) полей PUZZLE Opaque и Random #I перед расчетом подписи. Параметр HIP_SIGNATURE определён в параграфе 5.2.14, HIP_SIGNATURE_2 - в параграфе 5.2.15.

Расчёт HIP_SIGNATURE и HIP_SIGNATURE_2 выполняется как

```
HIP_SIGNATURE: { HIP header | [ Parameters ] }
```

где Parameters включает параметры пакета HIP с типами от 1 до (тип HIP_SIGNATURE - 1).

При расчёте подписи применяются два правила:

- в заголовке HIP устанавливается Checksum = 0;
- в заголовке HIP отсчёт поля Header Length начинается от начала параметра HIP_SIGNATURE.

Порядок параметров описан в параграфе 5.2.1.

```
HIP_SIGNATURE_2: { HIP header | [ Parameters ] }
```

где Parameters включает параметры пакета HIP с типами от 1 до (тип HIP_SIGNATURE_2 - 1).

При расчёте подписи применяются три правила:

- в заголовке HIP устанавливается Checksum = 0 и HIT = 0 (для Ответчика).
- в заголовке HIP отсчёт поля Header Length начинается от начала параметра HIP_SIGNATURE_2;
- В параметре PUZZLE для полей Opaque и Random #I устанавливается значение 0.

Порядок параметров описан в параграфе 5.2.1.

Процесс расчета и проверки подписей (обеих, за исключением случаев, где явно указано HIP_SIGNATURE_2) показан ниже.

Отправитель пакета

1. Создаётся пакет HIP без параметра HIP_SIGNATURE и следующих за ним параметров.
2. Рассчитывается поле Length и устанавливается Checksum = 0 в заголовке HIP. Для HIP_SIGNATURE_2 устанавливается HIT Инициатора в заголовке HIP, а также поля Opaque и Random #I в параметре PUZZLE.
3. Рассчитывается подпись с использованием секретного ключа, соответствующего HI (открытый ключ).
4. В пакет добавляется параметр HIP_SIGNATURE.
5. Добавляются параметры, следующие за HIP_SIGNATURE.
6. Пересчитывается поле Length в заголовке HIP и рассчитывается поле Checksum.

Получатель пакета

1. Проверяется поле Length и контрольная сумма в заголовке HIP.
2. Сохраняется содержимое HIP_SIGNATURE и следующих за ним параметров, после чего они удаляются.
3. Пересчитывается поле Length в заголовке HIP и устанавливается Checksum = 0. Для HIP_SIGNATURE_2 устанавливается HIT Инициатора в заголовке HIP, а также 0 для Opaque и Random #I в параметре PUZZLE.
4. Вычисляется подпись и сравнивается с полученной с использованием HI (открытй ключ) отправителя пакета.
5. Восстанавливается исходный пакет путём добавления удалённых параметров (п. 2) и восстановления сброшенных в 0 значений (п. 3).

При проверке может использоваться идентификатор HI из пакета HIP, HI полученный с помощью запроса DNS, если в параметре HOST_ID получено имя FQDN, или HI из иного источника.

6.5. Генерация HIP KEYMAT

Ключевой материал HIP выводится из сеансового ключа DH (Kij), создаваемого в процессе базового обмена HIP (см. параграф 4.1.3). Инициатор имеет ключ Kij при создании пакета I2, а Ответчик получает Kij в пакете I2, поэтому пакет I2 уже может содержать шифрованную информацию.

KEYMAT выводится путём подачи Kij в функцию вывода ключей, указанную DH Group ID. Этот документ определяет единственную функцию вывода ключей на основе хэширования (Hash-based Key Derivation Function или HKDF) [RFC5869] с использованием функции RHASH. Новые DH Group ID и соответствующие функции распределения ключей могут определять другие документы.

Ниже приведено более подробное описание вывода ключевого материала с использованием HKDF. Пусть

```
info    = sort(HIT-I | HIT-R)
salt    =  #I | #J
```

Sort(HIT-I | HIT-R) определяется как конкатенация двух тегов HIT в сетевом порядке байтов, при этом меньшее значение HIT предшествует большему. Теги сравниваются как положительные (без знака) 128-битовые целые числа с сетевым порядком байтов. Значения #I и #J берутся из головоломки и её решения, которые были переданы в сообщениях R1 и I2 при создании ассоциации HIP. Оба хоста сохраняют значения #I и #J в ассоциации HIP для применения в будущем.

Исходные ключи выводятся последовательно в порядке, определяемом численным сравнением тегов HIT, как указано выше. HOST_g указывает хост с большим значением HIT, HOST_l - с меньшим. Ниже показан порядок вывода четырёх начальных ключей.

Ключ шифрования HIP-gl для параметра ENCRYPTED хоста HOST_g.
Ключ защиты целостности (HMAC) HIP-gl для исходящих пакетов HIP хоста HOST_g.
Ключ шифрования HIP-lg для параметра ENCRYPTED хоста HOST_l.
Ключ защиты целостности (HMAC) HIP-lg для исходящих пакетов HIP хоста HOST_l.

Число битов, выведенных для данного алгоритма является «естественным» размером ключей. Для обязательных алгоритмов применяются указанные ниже размеры.

AES	128 или 256 битов
SHA-1	160 битов
SHA-256	256 битов
SHA-384	384 бита
NULL	0 битов

При использовании других размеров ключей они **должны** трактоваться как ключи для других алгоритмов и определяться отдельно.

6.6. Инициирование базового обмена HIP

Реализация может запустить базовый обмен HIP с другим хостом на основе решения локальной политики, обычно вызываемого дейтаграммой приложения, подобно обмену ключами IPsec IKE при динамической организации SA. Кроме того, система может инициировать обмен HIP после перезагрузки, тайм-аута или потери состояния HIP (параграф 4.5.4).

Реализация готовит пакет I1 и передаёт его по IP-адресу партнёра, который может быть получен традиционным способом, например, через DNS. Содержимое пакета I1 описано в параграфе 5.3.1. Выбор используемых Host Identity для Инициатора и Ответчика, если их несколько, обычно определяет политика. Ниже приведены концептуальные правила запуска базового обмена HIP.

1. Инициатор получает один или несколько тегов HIT Ответчика и один или несколько адресов с помощью запроса DNS по FQDN Ответчика, из иного репозитория или локальной базы данных. Если Инициатор не знает HIT Ответчика, он может попытаться использовать значение NULL в качестве HIT Ответчика (см. параграф 4.1.8. Режим HIP Opportunistic). Если у Инициатора есть выбор из нескольких HIT Ответчика, он выбирает HIT, для которого поддерживает HIT Suite.
2. Инициатор передаёт пакет I1 по одному из адресов Ответчика в соответствии с локальной политикой.
3. Инициатор включает в пакет I1 список DH_GROUP_LIST. Выбор и порядок DH Group ID в списке **должен** быть сохранен Инициатором, поскольку это потребуется при обработке пакета R1. В большинстве случаев предпочтения для групп DH будут статическими и не требуется хранить их для каждой ассоциации.
4. После передачи пакета I1 отправитель переходит в состояние I1-SENT и запускает таймер, значение которого **следует** выбирать больше наихудшего ожидаемого RTT. Отправителю также **следует** увеличить на 1 счётчик попыток, связанный с I1.
5. По завершению отсчёта таймера отправителю **следует** повторить пакет и заново запустить таймер, повторяя это до I1_RETRIES_MAX раз.

6.6.1. Параллельная отправка нескольких пакетов I1

Для минимизации задержки при создании ассоциации реализация **может** передать один и тот же пакет I1 по нескольким адресам Ответчика, однако **недопустимо** параллельно отправлять пакет более чем по трём (3) адресам Ответчика. Кроме того, при повторе по тайм-ауту реализация **должна** воздерживаться от передачи одного пакета I1 по нескольким адресам, т. е. при повторе инициализации соединения по тайм-ауту **недопустимо** передавать пакет I1 по нескольким адресам получателя. Эти ограничения предназначены для предотвращения перегрузки сети и возможных DoS-атак, например, когда кто-то заявил сотни или тысячи адресов, что может породить огромное число пакетов от Инициатора.

Поскольку не гарантируется, что Ответчик различит дубликаты пакетов I1, полученные по разным адресам (он не поддерживает состояния при отправке пакетов R1), Инициатор может получить несколько пакетов R1.

Инициатору **следует** выбрать предпочтительный адрес получателя по адресам отправителя в принятых пакетах R1 и использовать свой предпочтительный адрес в качестве источника для пакета I2. Правила обработки R1 даны в параграфе 6.8.

6.6.2. Обработка входящих сообщений ICMP Protocol Unreachable

Хост может получить сообщение ICMP Destination Protocol Unreachable в ответ на пакет HIP I1. Это может говорить о том, что партнёр не поддерживает HIP или указывать попытку атаки с целью обмануть Инициатора.

При получении системой сообщения ICMP Destination Protocol Unreachable в процессе ожидания пакета R1 **недопустимо** прерывать ожидание. **Можно** игнорировать сообщение ICMP и передать ещё несколько пакетов I1, а также **можно** счетсть сообщение ICMP намеком на то, что партнёр возможно не поддерживает HIP и вернуться в состояние UNASSOCIATED раньше, чем в ином случае. Однако хост **должен**, как минимум, продолжить ожидание пакета R1 в течение разумного времени до перехода в состояние UNASSOCIATED.

6.7. Обработка входящих пакетов I1

Реализации **следует** отвечать на I1 пакетом R1, за исключением случаев, когда она не хочет или не способна организовать ассоциацию HIP. Если реализация не может создать ассоциацию HIP, хосту **следует** передать сообщение «ICMP Destination Protocol Unreachable, Administratively Prohibited» по IP-адресу отправителя пакета I1. Если реализация не хочет создавать ассоциацию HIP, хост **может** игнорировать пакет I1. Это может быть, например, в случае DoS-атаки с пакетами I1.

Реализации **следует** поддерживать возможность обработки шквала принятых пакетов I1 с отбрасыванием дубликатов, полученных в коротком интервале времени. Обманные пакеты I1 могут вызывать атаку на систему с помощью пакетов R1, поэтому для R1 отправитель **должен** иметь механизм ограничения скорости отправки R1 по одному адресу. Конечному автомату HIP **рекомендуется** не менять состояние по факту отправки пакета R1.

Ниже приведены концептуальные правила отклика на пакет I1.

1. Ответчик **должен** убедиться, что HIT Ответчика в полученном пакете I1 совпадает с одним из его тегов HIT или имеет значение NULL. В противном случае он должен отбросить пакет.
2. Если Ответчик находится в состоянии ESTABLISHED, он **может** ответить пакетом R1, приготовиться к отбрасыванию имеющейся защищённой связи HIP с партнёром и сохранить состояние ESTABLISHED.
3. Если Ответчик находится в состоянии I1-SENT, он **должен** сравнить HIT отправителя со своим (получатель) HIT. Если HIT отправителя больше, следует отбросить пакет I1 и сохранить состояние I1-SENT. Если HIT отправителя меньше, **следует** передать пакет R1 и сохранить состояние I1-SENT. Сравнение HIT описано в параграфе 6.5.
4. Если реализация принимает решение ответить на I1 пакетом R1, она создаёт новый пакет R1 или выбирает заранее созданный, как описано в параграфе 5.3.2. Создаётся или выбирается пакет R1 с наиболее предпочтительным открытым ключом DH, который также включён в DH_GROUP_LIST пакета I1. Если нет подходящего DH Group ID в DH_GROUP_LIST пакета I1, пакет R1 передается с любым подходящим открытым ключом DH.
5. Если в пакете I1 тег HIT Ответчика имеет значение NULL, Ответчик выбирает HIT с тем же HIT Suite как у HIT Инициатора. Если Ответчик не поддерживает такой HIT Suite, ему **следует** выбрать **требуемый** HIT Suite (параграф 5.2.10), которым в настоящее время служит RSA/DSA/SHA-256. В остальном выбор HIT определяет локальная политика.
6. Ответчик указывает поддерживаемые форматы транспорта HIP в списке TRANSPORT_FORMAT_LIST, как указано в параграфе 5.2.11. Ответчик **должен** указать по меньше мере один формат транспорта для данных.
7. Ответчик передаёт пакет R1 по IP-адресу отправителя пакета I1.

6.7.1. Управление пакетами R1

Все совместимые реализации **должны** поддерживать создание пакетов R1. Даже если устройство настроено лиши на инициирование ассоциаций, оно должно быть способно обрабатывать пакеты I1 в случаях восстановления при потере состояния или завершении срока действия ключа. Пакеты R1 **можно** создавать заранее. Пакет R1 **можно** использовать повторно в течение короткого интервала Delta T, зависящего от реализации, и использованные пакеты **следует** отменять и не применять больше после получения действительного пакета I2 от Инициатора. При шквале пакетов I1 пакеты R1 **можно** применять за пределами обычного Delta T. Сведения R1 **недопустимо** отбрасывать до завершения интервале Delta S (зависит от реализации) после того, как пакет R1 больше не предлагается. Предполагается, что Delta S - это максимальное время, требуемое для того, чтобы последний пакет I2 в ответ на R1 прибыл к Ответчику.

Реализации, поддерживающие несколько групп DH, **могут** заранее создавать R1 для каждой поддерживаемой группы, чтобы быстро обрабатывать входящие пакеты I1 с разными DH Group ID в DH_GROUP_LIST.

Реализация **может** сохранять сведения о полученных пакетах I1 и сопоставлять с ними пакеты I2 (см. параграф 4.1.1).

6.7.2. Обработка сообщений с некорректным форматом

Если реализация получает пакет I1 с ошибками формата, ей **не следует** отвечать сообщением NOTIFY, поскольку это откпывает возможность для DoS-атак. Вместо этого можно ответить пакетов ICMP, как указано в параграфе 5.4.

6.8. Обработка входящих пакетов R1

Получившая пакет R1 система **должна** сначала проверить, отправляла ли она его источнику пакет I1 (состояние I1-SENT). Если этот так, пакет R1 **следует** обработать, как указано ниже, передать пакет I2 и перейти в состояние I2-SENT, установив таймер для защиты I2. Если система находится в состоянии I2-SENT, она **может** ответить на пакет R1, если он имеет большее значение счётчика генерации R1. В этом случае системе следует отбросить состояние, созданное в результате обработки прежнего пакета R1 и вернуться в состояние I1-SENT. При любом другом состоянии системы относительно передавшего пакет R1 хоста, пакет R1 **следует** просто отбросить.

При отправке нескольких пакетов I1 Инициатору **следует** подождать некоторое время после приёма первого пакета R1, чтобы могли прийти другие пакеты R1, а отвечать **следует** на пакет R1 с наибольшим значением счётчика генерации.

Ниже приведены концептуальные правила отклика на пакет R1.

1. Получившая пакет R1 система **должна** сначала проверить, отправляла ли она его источнику пакет I1 (т. е. наличие ассоциации HIP в состоянии I1-SENT, связанной с HIT из R1). Если пакет I1 не был передан в режиме Opportunistic mode (см. параграф 4.1.8), адрес IP из полученного пакета R1 **следует** игнорировать при обработке R1 и при поиске ассоциации HIP полученный пакет R1 **следует** сопоставлять лишь с ассоциациями, использующими HIT. При наличии соответствия системе следует обработать пакет R1, как указано ниже.
2. В ином случае, если состояние системы отличается от I1-SENT и I2-SENT относительно HIT в пакете R1, ей **следует** просто отбросить пакет R1 и сохранить текущее состояние.
3. Если ассоциация HIP имеет состояние I1-SENT или I2-SENT, полученный тег HIT Инициатора **должен** совпадать с HIT в исходном пакете I1. Кроме того, HIT Ответчика **должен** совпадать с указанным в I1, если там не было NULL HIT.
4. Системе **следует** проверить подпись R1, как указано в параграфе 5.2.15, до обработки пакета.
5. Если ассоциация HIP имеет состояние I1-SENT и имеется несколько действительных пакетов R1, система **должна** выбрать из них пакет с наибольшим значением счётчика генерации R1.
6. Система **должна** проверить наличие HIT Suite Инициатора в параметре HIT_SUITE_LIST пакета R1 (поддержка HIT Suite Ответчиком). Если Ответчик поддерживает HIT Suite, обработка продолжается нормально, в противном случае система **может** сохранить состояние I1-SENT и повторно запустить BEX, передав новый пакет I1 с HIT Инициатора, поддерживаемым Ответчиком и, следовательно, включённым в HIT_SUITE_LIST пакета R1. Система **может** прервать BEX, если нет подходящего HIT источника. Системе **следует** подождать

некоторое время для получения других пакетов R1 с большим значением счётчика генерации или иными HIT и HIT Suite, прежде чем перезапустить или прервать BEX.

7. Система **должна** убедиться, что DH Group ID в параметре DIFFIE_HELLMAN сообщения R1 совпадает с первым DH Group ID в DH_GROUP_LIST Ответчика в пакете R1, а также соответствует значению, включенному Инициатором в список DH_GROUP_LIST пакета I1. Если DH Group ID из параметра DIFFIE_HELLMAN не соответствует наилучшему выбору Ответчика, Инициатор может счесть, что список DH_GROUP_LIST в пакете I1 был намеренно изменен. В таком случае Инициатор **может** передать новый пакет I1, однако ему **не следует** менять предпочтения в DH_GROUP_LIST нового пакета I1. Инициатор **может** также прервать обмен HIP.
8. Если ассоциация HIP имеет состояние I2-SENT, система **может** вернуться в I1-SENT и обработать принятый пакет R1, если значение счётчика генерации R1 в нем больше, чем в ранее обработанном пакете R1.
9. В пакете R1 может быть установлен бит A и в этом случае система **может** отвергнуть (отбросить) пакет R1 и вернуться в состояние UNASSOCIATED. Системе **следует** рассматривать отбрасывание R1 лишь в случае использования ею NULL HIT в пакете I1. Если бит A установлен, HIT Ответчика является анонимным и его **не следует** хранить постоянно.
10. Системе **следует** попытаться сравнить HIT с полученным Host Identity, используя Host Identity для создания HIT и сравнения с HIT отправителя.
11. Система **должна** сохранить полученное значение счётчика генерации R1 для последующих операций.
12. Система пытается решить головоломку из пакета R1. Поиск решения **должен** прерываться по истечении срока действия головоломки. Если найти решение не удалось, реализация **может** повторить передачу I1 с учётом ограничений или прервать базовый обмен HIP.
13. Система рассчитывает стандартный ключевой материал Diffie-Hellman в соответствии с открытым значением и Group ID из параметра DIFFIE_HELLMAN. Ключевой материал DH Kij служит для извлечения ключей в соответствии с параграфом 6.5.
14. Система выбирает HIP_CIPHER ID из вариантов, предложенных в пакете R1, и использует выбранные значения при создании и применении ключей шифрования, а также при отправке пакета I2. Если предложенные варианты не подходят системе, она может повторить отправку I1 или прервать обмен HIP.
15. Система выбирает подходящий формат транспорта из списка TRANSPORT_FORMAT_LIST и включает соответствующий параметр транспортного формата в последующий пакет I2.
16. Система инициализирует оставшиеся переменные в соответствующем состоянии, включая счётчики Update ID.
17. Система готовит и передаёт пакет I2, как указано в параграфе 5.3.3.
18. Системе **следует** запустить таймер, для которого **следует** устанавливать значение больше худшего ожидаемого RTT, и она **должна** инкрементировать счётчик попыток, связанных с I2. Отправителю **следует** повторить пакет I2 по завершении отсчёта таймера и перезапустить таймер вплоть до I2_RETRIES_MAX раз.
19. Если система имеет состояние I1-SENT, ей **нужно** перейти в I2-SENT, в иных случаях состояние не меняется.

6.8.1. Обработка сообщений с некорректным форматом

При получении пакета R1 с некорректным форматом реализация должна отбросить его. Отправка NOTIFY или ICMP не поможет, поскольку отправитель пакета R1 обычно не имеет состояния. Реализации **следует** подождать возможного получения корректного пакета R1 и **можно** начать заново, передав новый пакет I1.

6.9. Обработка входящих пакетов I2

При получении пакета I2 система **может** выполнить начальную проверку соответствия пакета I2 переданному ранее пакету R1, если Ответчик хранит такое состояние. Например, отправитель может проверить, отправлен ли пакет I2 с адреса или HIT, откуда Ответчик недавно получил пакет I1. Пакет R1 может включать необрабатываемые (opaque) данные, которые возвращаются в пакете I2. Если пакет I2 кажется подозрительным, система **может** отбросить его.

Затем реализации **следует** обработать пакет I2. Это включает проверку решения головоломки, создание ключа DH, возможно расшифровку Host Identity Инициатора, проверку подписи и, наконец, отправку пакета R2.

Ниже приведены концептуальные правила отклика на пакет I2.

1. Система **может** выполнить проверки, чтобы убедиться в соответствии I2 недавно переданному пакету R1. Провирки зависят от реализации, пример представлен в Приложении A.
2. Система **должна** убедиться, что HIT Ответчика соответствует одному из его тегов HIT и должна отбросить пакет при несоответствии.
3. Система **должна** убедиться в поддержке HIT Инициатора. Ответчику **следует** отбрасывать пакеты I2 с неподдерживаемыми HIT Инициатора.
4. Если конечный автомат системы находится в состоянии R2-SENT, система **может** проверить сходство полученного пакета I2 с пакетом, вызвавшим переход в R2-SENT. Если пакеты похожи, система **может** повторить переданный ранее пакет R2 и сбросить таймер R2-SENT, сохранив состояние R2-SENT.
5. Если конечный автомат системы находится в состоянии I2-SENT, система **должна** сравнить локальный тег HIT с тегом отправителя (подобно описанию в параграфе 6.5). Если локальный тег HIT меньше HIT отправителя, следует отбросить пакет I2, использовать партнерский ключ DH и #I из полученного ранее пакета R1, а также взять локальный ключ DH и #J из переданного ранее пакета I2. В ином случае системе следует обработать полученный пакет I2 и отбросить ключевой материал DH Kij, который могу быть выведен при отправке предыдущего пакета I2. Партнерский ключ DH и #J берутся из свежепринятого пакета I2, локальный ключ DH и #I - из переданного ранее пакета R1.

6. Если конечный автомат системы находится в состоянии I1-SENT и теги HIT а пакете I2 соответствуют указанным в переданном ранее пакете I1, система использует принятый пакет I2 как основу для создаваемой ассоциации HIP и прекращает отправку пакетов I1 (при условии прохождения указанных ниже проверок I2).
7. Если конечный автомат системы находится в состоянии, отличном от R2-SENT, системе **следует** проверить попадание значения счётчика генерации R1, возвращённого в I2 (при наличии), в допустимый диапазон. Реализации **должны** воспринимать головоломки из текущего поколения и **могут** воспринимать более ранние. Если значение счётчика генерации в пакете I2 выходит за допустимые пределы, пакет I2 является устаревшим (возможно повторно использованным и его **следует** отбросить.
8. Система **должна** проверить решение головоломки (параграф 5.3.3), рассчитав хэш по алгоритму RHASH.
9. Пакет I2 **должен** иметь 1 значение в параметре HIP_CIPHER, которое **должно** совпадать с одним из значений, предложенных Инициатором в пакете R1.
10. Система должна вывести ключевой материал DH Kij на основе открытого значения и Group ID в параметре DIFFIE_HELLMAN. Этот материал применяется для создания ключей ассоциации HIP, как указано в параграфе 6.5. Если DH Group ID не поддерживается, пакет I2 просто отбрасывается.
11. Зашифрованный идентификатор HOST_ID расшифровывается с ключом Инициатора, определенным в параграфе 6.5. Если расшированные данные отличаются от HOST_ID, пакет I2 просто отбрасывается.
12. Реализации **следует** также проверить соответствие HIT Инициатора в пакете I2 значению Host Identity, переданному в пакете I2 (некоторые промежуточные устройства не способны выполнить такую проверку).
13. Система **должна** обработать параметр TRANSPORT_FORMAT_LIST. Спецификацию обработки транспортных параметров задают отдельные документы (например, [RFC7402]).
14. Система **должна** проверить HIP_MAC в соответствии с процедурами параграфа 5.2.12.
15. Система **должна** проверить HIP_SIGNATURE в соответствии с параграфами 5.2.14 и 5.3.3.
16. Если указанные выше проверки прошли, система продолжает обработку I2, в противном случае она отбрасывает пакет I2 и не меняет состояние конечного автомата.
17. В пакете I2 может быть установлен бит A и в этом случае система **может** отвергнуть (отбросить) пакет I2, возвратив конечный автомат в состояние UNASSOCIATED. При установленном бите A тег HIT Инициатора не следует хранить постоянно.
18. Система инициализирует оставшиеся переменные в соответствующем состоянии, включая счётчики Update ID.
19. Если после успешной обработки I2 конечный автомат находится в состоянии UNASSOCIATED, I1-SENT, I2-SENT или R2-SENT, передаётся пакет R2 и конечный автомат переходит в состояние R2-SENT.
20. Если после успешной обработки I2 конечный автомат находится в состоянии ESTABLISHED, старая ассоциация HIP сбрасывается и создаётся новая, передаётся пакет R2 и конечный автомат переходит в состояние R2-SENT.
21. При переходе конечного автомата в состояние R2-SENT система запускает таймер. Состояние конечного автомата меняется на ESTABLISHED, если через входящую ассоциацию HIP получены данные или принят пакет UPDATE (или иной пакет, указывающий переход удалённого партнёра в состояние ESTABLISHED). По завершении отсчёта таймера (с учётом максимального числа повторов I2) конечный автомат переходит в состояние ESTABLISHED.

6.9.1. Обработка сообщений с некорректным форматом

Поведению реализации, получившей пакет I2 с некорректным форматом, следует зависить от числа уже прошедших проверок. Если решение головоломки в пакете оказалось верным, реализации **следует** сообщить об ошибке пакетом NOTIFY. В ином случае реализация **может** ответить сообщением ICMP, как указано в параграфе 5.4.

6.10. Обработка входящих пакетов R2

Пакет R2, полученный в состоянии UNASSOCIATED, I1-SENT или ESTABLISHED будет отбрасываться без смены состояния конечного автомата. В состоянии I2-SENT пакет R2 **должен** обрабатываться.

Ниже приведены концептуальные правила отклика на пакет R2.

1. Если состояние системы отличается от I2-SENT, пакет R2 просто отбрасывается.
2. Система **должна** убедиться, что применяемые теги HIT соответствуют полученным в пакете R1, вызвавшем переход в состояние I1-SENT.
3. Система **должна** проверить HIP_MAC_2 в соответствии с процедурами параграфа 5.2.13.
4. Система **должна** проверить подпись HIP в соответствии с процедурами параграфа 5.2.14.
5. Отказ при любой из указанных выше проверок с высокой вероятностью говорит о MitM или иной атаке. Системе **следует** вести себя подобающим образом в соответствии с локальной политикой.
6. После успешной обработки пакета R2 конечный автомат переходит в состояние ESTABLISHED.

6.11. Передача пакетов UPDATE

Хост передаёт пакет UPDATE для обновления сведений, связанных с ассоциацией HIP. Возможно много вариантов применения этого пакета, например, управление мобильностью или смена ключей в имеющейся ESP SA. Ниже описаны концептуальные правила передачи пакетов UPDATE партнеру. В других документах моугт быть определены дополнительные применения пакетов UPDATE.

Последовательность пакетов UPDATE указывается параметром SEQ. Перед отправкой UPDATE система сначала проверяет, не осталось ли незавершенный пакетов UPDATE, которые могут помешать обработке нового пакета. Когда обработки ожидает (не подтверждены) несколько пакетов UPDATE, отправитель должен считать, что получатель может обрабатывать их в произвольном порядке. Поэтому новые пакеты UPDATE, зависящие от приёма и обработки предыдущих UPDATE, **должны** откладываться до получения требуемых подтверждений ACK. Одним из способов предотвращения конфликтов является разрешение иметь лишь один необработанный пакет UPDATE в каждый момент времени. Однако разрешение нескольких UPDATE может повысить производительность мобильных и многоадресных (многодомных) протоколов.

Ниже приведены концептуальные правила передачи пакетов UPDATE.

1. Первый пакет UPDATE передаётся с Update ID = 0. В иных случаях система сама устанавливает Update ID, увеличивая значение в каждом переданном пакете.
2. Система создаёт пакет UPDATE с параметром SEQ, содержащим текущее значение Update ID. Пакет может также включать параметры ACK, подтверждающие Update ID из предыдущих UPDATE SEQ от партнера.
3. Система передаёт созданное сообщение UPDATE и запускает таймер UPDATE, для которого по умолчанию устанавливается удвоенное значение оценки RTT. При наличии нескольких необработанных сообщений UPDATE работает несколько таймеров.
4. По завершении отсчёта таймера пакет UPDATE передаётся заново до UPDATE_RETRY_MAX раз. Для последующих повторов таймер UPDATE **следует** экспоненциально увеличивать. Если после UPDATE_RETRY_MAX попыток не было получено подтверждения, ассоциация HIP считается разорванной и конечный автомат **следует** перевести из состояния ESTABLISHED в CLOSING, как указано в параграфе 4.4.4. Таймер UPDATE останавливается при получении от партнёра ACK с подтверждением UPDATE.

6.12. Приём пакетов UPDATE

Обработка полученного системой пакета UPDATE зависит от состояния ассоциации HIP, а также наличия и значений параметров SEQ и ACK. Обычно сообщения UPDATE содержать также необязательные параметры, обработка которых описана в отдельных документах.

Для каждой ассоциации хост сохраняет следующий ожидаемый идентификатор обновления от партнёра (peer Update ID), начальное значение которого равно 0. Сравнение номеров (больше, меньше) выполняется в циклическом пространстве номеров, поэтому **должна** выполняться соответствующая обработка перехода через максимум (2^{32}).

Отправитель **может** передать несколько остающихся сообщений UPDATE и они обрабатываются в полядке их приёма получателем (т. е. без восстановления порядка). При обработке сообщений UPDATE без соблюдения порядка получатель **должен** отслеживать, какие сообщения UPDATE были обработаны ранее, чтобы дубликаты и повторы пожтверждались без дополнительной обработки. Получатель **может** задать окно приёма для Update ID, которые он желает обрабатывать в данный момент и отбрасывать сообщения UPDATE, не попадающие в это окно.

Ниже приведены концептуальные правила обработки принимаемых пакетов UPDATE.

1. При отсутствии соответствующей ассоциации HIP реализация **может** ответить сообщением ICMP Parameter Problem, как указано в параграфе 5.4.4.
2. Если ассоциация находится в состоянии ESTABLISHED и присутствует параметр SEQ (но не ACK), обработка UPDATE и отклик на пакет выполняются в соответствии с параграфом 6.12.1.
3. Если ассоциация находится в состоянии ESTABLISHED и присутствует параметр ACK (но не SEQ), обработка UPDATE выполняется в соответствии с параграфом 6.12.2.
4. Если ассоциация находится в состоянии ESTABLISHED и в UPDATE присутствуют параметры ACK и SEQ, сначала обрабатывается ACK в соответствии с параграфом 6.12.2, затем - остальная часть UPDATE в соответствии с параграфом 6.12.1.

6.12.1. Обработка параметра SEQ в принятом пакете UPDATE

Ниже приведены концептуальные правила обработки параметра SEQ в принятом пакете UPDATE.

1. Если Update ID в принятом SEQ не является следующим в последовательности Update ID и превышает верхнюю границу окна приёма для новых UPDATE, пакет **должен** отбрасываться.
2. Если Update ID в принятом SEQ соответствует недавно обработанному UPDATE, пакет считается повторным. Проверку HIP_MAC (следующий шаг) **недопустимо** пропускать (хотя приемлемо побайтовое сравнение полученного пакета с сохраненным). Хосту рекомендуется кэшировать пакеты UPDATE, переданные с ACK, для предотвращения издержек, связанных с созданием нового пакета ACK в ответ на повторный пакет UPDATE. Система **должна** снова подтверждать (очевидные) повторы UPDATE, но **следует** также учитывать ограничение частоты откликов на повторы для защиты от replay-атак.
3. Система **должна** проверить HIP_MAC в пакете UPDATE и при несовпадении **должна** отбросить пакет.
4. Система **может** проверить SIGNATURE в пакете UPDATE. При несовпадении пакет **следует** отбросить, записав в журнал сообщение об ошибке.
5. Если обрабатывается новый параметр SEQ, после этого обрабатываются параметры UPDATE. Система **должна** записать Update ID из принятого параметра SEQ для защиты от повторного использования.
6. Подготавливается и передаётся партнёру пакет подтверждения UPDATE с параметром ACK. Этот параметр **может** включаться в отдельный пакет UPDATE или присоединяться к UPDATE с параметром SEQ, как указано в параграфе 5.3.5. Параметр ACK **может** подтверждать несколько Update ID от партнера.

6.12.2. Обработка параметра ACK в принятом пакете UPDATE

Ниже приведены концептуальные правила обработки параметра ACK в принятом пакете UPDATE.

1. Порядковый номер в ACK должен соответствовать переданному ранее пакету UPDATE, который ещё не подтвержден. Если совпадения не найдено или ACK не подтверждает новый пакет UPDATE, тогда **должен** быть отброшен прежний пакет (при отсутствии параметра SEQ) или выполнена обработка в соответствии с параграфом 6.12.1.
2. Система **должна** проверить HIP_MAC в пакете UPDATE и при несовпадении **должна** отбросить пакет.
3. Система **может** проверить SIGNATURE в пакете UPDATE. При несовпадении пакет **следует** отбросить, записав в журнал сообщение об ошибке.
4. Соответствующий таймер UPDATE останавливается (см. параграф 6.11), чтобы подтверждённый сейчас пакет UPDATE больше не передавался. При подтверждении нескольких UPDATE останавливаются все соответствующий таймеры.

6.13. Обработка пакетов NOTIFY

Обработка пакетов NOTIFY **необязательна**. Если обработка выполняется, все ошибки в полученных параметрах NOTIFICATION **следует** записывать в системный журнал. Эти ошибки **должны** считаться лишь информацией и получателю **не следует** менять состояние HIP (см. параграф 4.4.2) лишь на основе полученного сообщения NOTIFY.

6.14. Обработка пакетов CLOSE

Получив сообщение CLOSE, хост отвечает сообщением CLOSE_ACK и переходит в состояние CLOSED (достоверность CLOSE проверяется с использованием HIP_MAC и SIGNATURE). Эта обработка выполняется независимо от нахождения ассоциации HIP в состоянии CLOSING, чтобы можно было обработать отправленные обеими сторонами сообщения CLOSE. Ассоциация HIP не сбрасывается до перехода хоста в состояние UNASSOCIATED.

После запуска процесса закрытия любая новая потребность в отправке данных вызывает создание новой ассоциации HIP, начиная с отправки пакета I1.

При отсутствии соответствующей ассоциации HIP пакет CLOSE отбрасывается.

6.15. Обработка пакетов CLOSE_ACK

При получении сообщения CLOSE_ACK хост проверяет, что он находится в состоянии CLOSING или CLOSED, а CLOSE_ACK является откликом на CLOSE. Хост может сопоставлять сообщения CLOSE_ACK с CLOSE, сравнивая значение ECHO_REQUEST_SIGNED (в пакете CLOSE) с ECHO_RESPONSE_SIGNED (в пакете CLOSE_ACK).

CLOSE_ACK содержит параметры HIP_MAC и SIGNATURE для проверки. Состояние сбрасывается при переходе в UNASSOCIATED и после этого хост **может** отвечать параметром ICMP Parameter Problem на входящие сообщения CLOSE (см. параграф 5.4.4).

6.16. Обработка потери состояния

В случае аварии или непредвиденной потери состояния системе **следует** удалить соответствующее состояние HIP, включая ключевой материал, т. е. состояние **не следует** помещать в долговременное хранилище. Если реализация отбрасывает состояние (как **рекомендуется**), она **должна** также сбросить значение счётчика генерации R1, если локальная политика явно не задаёт сохранение значения для конкретного хоста. реализации **недопустимо** сохранять по умолчанию партнерские значения счетчиков создания R1, но при сохранении этих значений это **должно** явно настраиваться в HIT.

7. Правила HIP

Имеется множество переменных, влияющих на базовый обмен HIP, которые должен поддерживать каждый хост. Все реализации HIP **должны** поддерживать одновременно более одного идентификатора HI, из которых хотя бы один **следует** резервировать для анонимного использования. Хотя анонимные HI редко применяются как HI Ответчика, они часто нужны для Инициаторов. **Рекомендуется** поддерживать более двух HI.

Инициаторы **могут** применять разные HI для разных Ответчиков с целью обеспечения базовой приватности. Применение таких приватных HI повторно для того же Ответчика и срок использования этих HI определяется локальной политикой и зависит от требований Инициатора к приватности.

Значение #K в HIP R1 следует выбирать с осторожностью. Слишком высокие значения #K будут мешать клиентам со слабым CPU, поскольку они не смогут решить головоломку за разумное время. Значение #K следует повышать лишь в том случае, когда Ответчик сильно загружен, т. е. больше не может обрабатывать все входящие согласования HIP. Если Ответчик не загружен сильно, **следует** устанавливать #K = 0.

Ответчикам, которые работают лишь с заданными Инициаторами, нужны списки управления доступом (Access Control List или ACL), указывающие хосты, от которых они воспринимают базовый обмен HIP, а также предпочтительный формат транспорта и локальный срок действия. В таких ACL **следует** поддерживать шаблоны, в том числе на общедоступных или предоставляющих анонимные услуги Ответчиках.

8. Вопросы безопасности

Протокол HIP предназначен для защищённой аутентификации хостов и также пытается ограничить раскрытие хостов для разных DoS- и MitM-атак. Однако сам протокол HIP может подвергаться таким атакам, которые способны повредить обычной работе хостов.

В DoS-атаках часто используется преимущество асимметрии издержек на создание ассоциации. Одним из примеров такой асимметрии является необходимость сохранения Ответчиком локального состояния при возможности

Инициатора обходиться без установки состояния. HIP не пытается повысить издержки при создании ассоциации для Инициатора, но пытается снизить их для Ответчика. Это достигается за счёт того, что Ответчик запускает трехэтапный обмен вместо Инициатора, что делает обмен HIP 4-пакетным. При этом первый пакет Ответчика (R1) может быть подготовлен заранее и Ответчик **может** применять заготовленные пакеты многократно, пока тот или иной Инициатор не предоставит действительный отклик на такой пакет R1. В случае лавины потоков I1 хост может многократно применять одно значение DH, даже если какой-либо Инициатор предоставил действительный отклик с этим значением DH. Однако такое поведение не рекомендуется и его следует избегать. Использование одних значений Diffie-Hellman и случайного значения головоломки #I сопряжено с некоторым риском и нужно соблюдать баланс между таким риском и возможным шквалом пакетов HIP I1.

Смещение издержек создания состояния к Инициатору при создании пакетов HIP I2 представляет собой другую DoS-атаку. Злоумышленник может подделать пакет I1 и Ответчик передаст ему пакет HIP R1. Это позволяет предположительно связать Инициатора с оценкой пакета HIP R1 и созданием пакета I2. Защитой от таких атак является простое игнорирование пакетов R1, которые не соответствуют переданным I1 (параграф 6.8, п. 1).

Пакет R1 значительно больше пакета I1 и эту асимметрию можно использовать для атак с отражением. Злоумышленник может использовать IP-адрес жертвы для передачи шквала пакетов I1 мощному Ответчику, а тот для каждого небольшого пакета I1 будет передавать жертве более крупный пакет R1, что усиливает лавинную атаку с отражением. Для предотвращения таких атак Ответчику **следует** ограничивать частоту передачи пакетов R1 в целом или для конкретного адреса IP.

Другим типом DoS-атак является лавинная отправка обменных пакетов I2. Когда атакующий Инициатор решил головоломку, он может передавать пакеты с подставным IP-адресом отправителя, включающие недействительную подпись HIP или недействительные шифрованные данные HIP (в параметре ENCRYPTED). Это потребует от Ответчика расхода ресурсов для определения того, что пакет I2 не может быть обработан полностью. Защитой от таких атак является отбрасывание пакетов I2 с данным решением головоломки после приёма N некорректных пакетов I2. Это остановит атаку и злоумышленнику потербуется запросить другой пакет R1 для её продолжения. Ответчик может во время таких атак увеличивать значение #K. Для сохранения списка решений из некорректных пакетов Ответчику требуется сохранять состояние для этих пакетов I2, пока не увеличится значения счетчика R1. Поскольку некорректные пакеты обычно отфильтровываются по контрольной сумме до проверки подписи, в «черный список» помещаются лишь решения головоломки из пакетов, которые созданы для прохождения проверки контрольной суммы и головоломки. Кроме того, перед созданием новой записи в списке требуется верное решение головоломки, поэтому для лавинной атаки на «черный список» злоумышленнику придётся сначала решить головоломки.

Ещё одним вариантом DoS-атаки является имитация перезапуска состояния после перезагрузки одного из партнёров. Перезапускающийся хост передаёт партнёрам пакеты I1, на которые те будут отвечать пакетом R1 даже находясь в состоянии ESTABLISHED. Если пакет I1 является обманным, принятый пакет R1 будет неожиданным для хоста, чей адрес использован в обманном пакете, и будет отброшен как в описанном выше случае.

Четвёртый вариант DoS-атаки заключается в имитации разрыва ассоциации HIP. Протокол HIP полагается на таймеры и обмен CLOSE/CLOSE_ACK для явного сигнала о завершении ассоциации HIP. Поскольку сообщения CLOSE и CLOSE_ACK содержат HIP_MAC, посторонний не может закрыть соединение. Наличие параметра SIGNATURE позволяет промежуточным устройствам (например, межсетевым экранам, трансляторам NAT на основе SPI) просматривать эти сообщения и отбрасывать соответствующее состояние. Однако необязательный ответ на сообщение CLOSE пакетом ICMP Parameter Problem (см. параграф 5.4.4) может позволить атакующему подменить IP-адрес отправителя для отправки сообщений CLOSE, запускающих атаку с отражением.

Пятой формой DoS-атаки является повторное использование пакетов R1 с целью вынудить Инициатора решать устаревшие головоломки с потерей синхронизации с Ответчиком. Монотонно возрастающий счётчик R1 предназначен для защиты от таких атак (см. параграф 4.1.4).

Защититься от MitM-атак (перехват с участием человека) сложно без сторонней аутентификации. Опытный злоумышленник MitM может легко справиться со всеми частями HIP, однако протокол HIP опосредованно обеспечивает защиту от MitM-атак. Если HI Ответчика получен из подписанной зоны DNS, сертификата или иным защищённым способом, Инициатор может использовать его для проверки пакета HIP R1.

Аналогично, если HI Инициатора хранится в защищённой зоне DNS, доверенном сертификате или получен иным защищённым способом, Ответчик может получить HI (после получения пакета HIP I2) и проверить, можно ли доверять HI.

В этом документе представлена концепция режима HIP opportunistic, но документ не задаёт семантику такой организации соединения для приложений. Как отмечено в параграфе 4.1.8, имеются некоторые опасения по поводу режима opportunistic.

Сообщения NOTIFY применяются лишь с информационными целями и не подтверждаются. Реализация HIP не может полагаться лишь на сведения из NOTIFY, поскольку эти пакеты можно использовать повторно (replay). Реализации **не следует** менять какие-либо данные о состоянии на основе лишь полученного сообщения NOTIFY.

Поскольку протокол HIP будут поддерживать не все хосты, предполагается использование сообщений ICMP Destination Protocol Unreachable для организации DoS-атак. Атака на Инициатора будет выглядеть так, будто Ответчик не поддерживает HIP, но вскоре после приёма сообщения ICMP Инициатор получит действительный пакет HIP R1. Поэтому для защиты от таких атак Инициатору **не следует** реагировать на сообщение ICMP в течение некоторого времени, пока не будет получен реальный пакет HIP R1 от Ответчика. Аналогичная атака на Ответчик более сложна. Обычно, если принятое Ответчиком сообщение I1 является обманным сообщением от злоумышленника, Ответчик может получить сообщение ICMP с IP-адреса, по которому было передано сообщение R1. Однако изощренный злоумышленник может попытаться воспользоваться этим с целью прервать базовый обмен HIP с помощью передачи Ответчику такого сообщения ICMP до того, как Инициатор сможет отправить действительный пакет I2. Поэтому Ответчику **не следует** реагировать на такие сообщения ICMP. В частности, ему **не следует** удалять какое-либо минимальное состояние, созданное при отправке пакета HIP R1 (если оно было создано), а следует дождаться получения действительного пакета HIP I2 или естественного тайм-аута (если пакеты R1 отслеживаются). Точно так же Инициатору **следует** игнорировать любые сообщения ICMP в процессе ожидания пакета HIP R2 и **следует** удалять какие-либо ожидающие состояния лишь по естественному тайм-ауту.

9. Взаимодействие с IANA

Агентство IANA зарегистрировало номер протокола 139 для Host Identity Protocol и включило его в реестры IPv6 Extension Header Types [RFC7045] и Assigned Internet Protocol Numbers. Ссылка на [RFC5201] в обоих реестрах заменена ссылкой на данную спецификацию.

Ссылка на [RFC5201] для 128-битового значения 0xF0EF F02F BFF4 3D0F E793 0C3C 6E61 74EA в пространстве имён CGA Message Type [RFC3972] заменена ссылкой на данную спецификацию.

В реестр Host Identity Protocol (HIP) Parameters были внесены перечисленные ниже изменения. Многие изменения связаны с заменой ссылки на [RFC5201] ссылкой на данную спецификацию, но имеются и другие отличия. Терминология выделения значений определена в [RFC5226], имеющиеся упоминания IETF Consensus заменены на IETF Review в соответствии с [RFC5226].

HIP Version - версия HIP

Этот документ добавляет в реестр значение 2. Для значения 1 сохранена ссылка на [RFC5201].

Packet Type - тип пакета

7-битовое поле Packet Type в протоколе HIP описывает тип протокольного сообщения HIP. Поле определено в параграфе 5.1. Все имеющиеся ссылки на [RFC5201] заменены ссылками на эту спецификацию. Остальные значения сохранены.

HIT Suite ID - идентификатор набора HIT

Эта спецификация создаёт новый реестр HIT Suite ID. Он отличается от имеющегося реестра Suite ID, который сохранен для протокола версии 1 [RFC5201] (этот реестр закрыт для новых регистрация).

4-битовое значение HIT Suite ID использует поле OGA ID из ORCHID для указания типа HIT. Этот документ определяет три HIT Suite (см. параграф 5.2.10).

HIT Suite ID также передаётся в 4 старших битах поля ID в параметре HIT_SUITE_LIST, а 4 младших бита зарезервированы для будущих расширений пространства HIT Suite ID (сверх 16 значений).

В настоящее время HIT Suite использует лишь 4 бита, поскольку они передаются в HIT. Расширение числа битов в HIT Suite ID снижает криптостойкость HIT. Значения HIT Suite ID следует выделять с осторожностью для экономии пространства значений. Более того, отмененные идентификаторы следует выделять повторно по прошествии достаточного времени. Если 15 значений Suite ID (значение 0 исходно является резервным) окажется не достаточно, можно использовать большее число HIT Suite ID с помощью резервного значения HIT Suite ID = 0 для индикации большего числа битов. Параметр HIT_SUITE_LIST уже поддерживает 8-битовые HIT Suite ID, если они нужны. Однако [RFC7343] в настоящее время не поддерживает такое расширение. Предлагается сначала воспользоваться методом «опрокидывания» (rollover), описанным а Приложении E. Возможные расширения пространства HIT Suite ID до 8 битов и новые значения HIT Suite ID выделяются по процедуре IETF Review.

В запросы на регистрацию использовавшихся ранее значений следует включать примечание о повторном использовании значения после истечения определённого времени, чтобы обеспечить надлежащую проверку и одобрение IETF.

Parameter Type - тип параметра

16-битовое поле Type в параметре HIP описывает тип параметра (определено в параграфе 5.2.1). Текущие значения типов рассматриваются в параграфах 5.2.3 - 5.2.23. Реестр Parameter Types обновлён, как описано ниже.

Добавлено значение 129 для параметра R1_COUNTER со ссылкой на эту спецификацию, а имеющееся значение 128 для этого параметра сохранено со ссылкой на [RFC5201]. Это документирует изменение, внесенной в версии 2 для данного протокола. Для ясности имя значения 128 было сменено на R1_Counter (v1 only).

Добавлено значение 579 для нового типа HIP_CIPHER со ссылкой на эту спецификацию. Этот тип параметра функционально заменяет тип HIP_TRANSFORM (577), который сохранен со ссылкой на [RFC5201]. Для ясности имя значения 577 заменено на HIP_TRANSFORM (v1 only).

Добавлено значение 715 для HIT_SUITE_LIST со ссылкой на эту спецификацию.

Добавлено значение 2049 для TRANSPORT_FORMAT_LIST со ссылкой на эту спецификацию.

Имя типа HMAC (61505) заменено на HIP_MAC, а HMAC_2 (61569) - на HIP_MAC_2 со ссылками на эту спецификацию.

Для остальных типов со ссылками на [RFC5201] эти ссылки были обновлены с указанием данного документа, а типы со ссылками на иные RFC оставлены без изменений.

Типы 32768 - 49151 (не 49141 - значение, указанное в прежней таблице) были зарезервированы для приватного использования (Reserved for Private Use). Реализациям **следует** выбирать типы из этого диапазона случайно для снижения вероятности конфликтов. При этом **следует** использовать метод, обеспечивающий фактическую случайность (например, бросание монеты).

Указанная ранее процедура выделения в порядке запросов с требованием спецификации (First Come First Served with Specification Required) заменена на Specification Required (нужна спецификация).

Group ID - идентификатор группы

8-битовые значения Group ID применяются в параметрах DIFFIE_HELLMAN и DH_GROUP_LIST (параграф 5.2.7). Реестр обновлён с учётом новых значений, определённых в параграфе 5.2.7, устаревшие значения сохранены в реестре с пометкой DEPRECATED и ссылкой на [RFC5201]. Новые значения выделяются по процедуре IETF Review.

HIP Cipher ID - идентификатор шифра HIP

16-битовые значения Cipher ID в параметре HIP_CIPHER определены в параграфе 5.2.8. Это новый реестр. Значения из неиспользуемого и резервного пространства выделяются по процедуре IETF Review.

DI-Type - тип DI

4-битовые значения DI-Type в параметре HOST_ID определены в параграфе 5.2.9. Новые значения выделяются по процедуре IETF Review. Имеющиеся значения со ссылкой на [RFC5201] обновлены с указанием этой спецификации.

HI Algorithm - алгоритм HI

16-битовые значения Algorithm в параметре HOST_ID определены в параграфе 5.2.9. Это новый реестр. Значения из неиспользуемого и резервного пространства выделяются по процедуре IETF Review.

ECC Curve Label - метка кривой ECC

Когда для HI Algorithm в параметре HOST_ID определено значение ECDSA или ECDSA_LOW, нужен новый реестр для значений ECC Curve Label, определённых в параграфе 5.2.9. Это может быть сделано созданием двух

определяемых алгоритмом субреестров с именами ECDSA Curve Label и ECDSA_LOW Curve Label. Новые значения выделяются по процедуре IETF Review.

Notify Message Type - тип сообщения Notify

16-битовые значения Notify Message Type в параметре NOTIFICATION определены в параграфе 5.2.19.

Notify Message Type от 1 до 10 служат для информирования об ошибках в структуре пакета, 11-20 - о проблемах в параметрах, содержащих относящийся к криптографии материал, 21-30 - о проблемах при аутентификации или проверке целостности пакета. Типы выше 30 служат для информирования об иных типах ошибок или событиях.

Процедуры регистрации были обновлены. Для значений 1-50 сохраняется процедура IETF Review, для 51-8191 применяется процедура Specification Required. Для диапазона 8192-16383 сохранен статус Reserved for Private Use, для 16384-40959 применяется процедура Specification Required, а для 40960-65535 сохранен статус Reserved for Private Use.

В имеющийся реестр внесены указанные ниже изменения и ссылки на [RFC5201] заменены ссылками на эту спецификацию.

Тип INVALID_HIP_TRANSFORM_CHOSEN переименован в INVALID_HIP_CIPHER_CHOSEN с прежним значением 17.

Добавлено значение 20 для типа UNSUPPORTED_HIT_SUITE.

Тип HMAC_FAILED переименован в HIP_MAC_FAILED с прежним значением 28.

Тип SERVER_BUSY_PLEASE_RETRY переименован в RESPONDER_BUSY_PLEASE_RETRY с прежним значением 44.

10. Отличия от RFC 5201

В этом разделе приведена сводка технических изменений, внесенных в [RFC5201]. Раздел является информационным и предназначен для оказания помощи разработчикам при реализации новой версии протокола. Если текст этого раздела противоречит информации других разделов, нормативной следует считать информацию из других разделов.

Этот документ задаёт протокол HIP версии 2, который не совместим с HIP версии 1 [RFC5201]. Основными техническими изменениями являются включение дополнительных функций криптографической гибкости, а также обновление обязательных и дополнительных алгоритмов, включая поддержку эллиптических кривых с помощью алгоритмов Elliptic Curve DSA (ECDSA) и Elliptic Curve Diffie-Hellman (ECDH). Обновлены обязательные для реализации криптоалгоритмы, включая замену HMAC-SHA-1 на HMAC-SHA-256 и алгоритма подписи RSA/SHA-1 на RSASSA-PSS, а также добавление ECDSA к RSA в качестве обязательного типа открытых ключей. Эта версия HIP согласована с пересмотром ORCHID [RFC7343].

Ниже перечислены изменения, внесённые в работу протокола.

- Впараграфе 4.1.3 описан новый процесс для согласования группы Diffie-Hellman (аспект криптографической гибкости). Инициатор может указать предпочтения при выборе группы DH в пакете I1 и предложить несколько вариантов. Ответчик возвращает свои предпочтения на основе локальной политики и предложения Инициатора. Если предложенный Ответчиком выбор не устраивает Инициатора (неподдерживаемый алгоритм), он может запустить базовый обмен заново.
- Другим аспектом криптографической гибкости является добавления возможности применять разные криптографические хэш-функции для создания HIT. Для поддержки этого была введена терминология алгоритма хэширования для HIT Ответчика (RHASH). Кроме того, были добавлены HIT Suite для группировки набора криптографических алгоритмов, применяемых совместно для подписи открытого ключа, функции хэширования и отсечки хэш-значений. Применение HIT Suite ограничивает комбинации при выборе алгоритмов для различных функций. Идентификаторы HIT Suite ID связаны с полем ORCHID OGA ID ([RFC7343]).
- The puzzle mechanism has been slightly changed, in that the #I parameter depends on the HIT hash function (RHASH) selected, and the specification now advises against reusing the same #I value to the same Initiator; more details are provided in Sections 4.1.2 and 5.2.4).
- Параграф 4.1.4 расширен разъяснением деталей перехода счётчика генерации R1 через максимум и сброса.
- Добавлен параграф 4.1.6 с описанием процедур прерывания базового обмена HIP.
- В параграфе 4.1.7 даны рекомендации по предотвращению атак на снижение стойкости криптоалгоритмов.
- В параграфе 4.1.8 обновлено описание режима Opportunistic с учётом криптографической гибкости за счёт процедур выбора HIT.
- Обновлено описание генерации HIP KEYMAT в параграфе 6.5 с учётом согласования функции вывода ключей.
- Обновлено описание обработки пакетов I1, R1, I2 с учётом новых параметров.
- Добавлено требование, в соответствии с которым хосты должны поддерживать обработку параметров ACK с несколькими порядковыми номерами SEQ, даже если они не поддерживают передачу таких параметров.
- Разъяснено, что в пакете R1 может присутствовать несколько параметров ECHO_REQUEST_UNSIGNED, а в I2 - несколько параметров ECHO_RESPONSE.
- Добавлены процедуры отклика на несоответствие версия сообщениями ICMP Parameter Problem.
- Обновлён раздел 8. Вопросы безопасности с удалением атак, которые более не считаются возможными.
- Использование бита Anonymous для возможности применения анонимного Host Identity отправителя поддерживается теперь не только в пакетах R1 и I2.
- Использование шифра NULL HIP CIPHER явно ограничено отладкой и тестированием HIP и поддержка алгоритма перестала быть обязательной.

Ниже перечислены изменения, внесённые в типы и кодирование параметров (параграф 5.2).

- Добавлены 4 типа: DH_GROUP_LIST, HIP_CIPHER, HIT_SUITE_LIST, TRANSPORT_FORMAT_LIST.

- 2 типа параметров переименованы - HMAC в HIP_MAC и HMAC2 в HIP_MAC_2.
- 1 тип признан устаревшим - HIP_TRANSFORM. Функционально он заменён типом HIP_CIPHER, имеющим несколько иную семантику (хеш-значения удалены и определяются параметром RHASH).
- Параметр TRANSPORT_FORMAT_LIST позволяет согласовать транспорт со списком, а не по порядку размещения в пакет HIP.
- Код типа для R1_COUNTER сменен со 128 на 129, поскольку сейчас параметр является критическим и должен возвращаться при его наличии в R1.
- Размер параметров PUZZLE и SOLUTION стал переменным и зависит от размера RHASH.
- Обновлены поддерживаемые идентификаторы Diffie-Hellman Group ID.
- Параметр HOST_ID сейчас требует указания Algorithm.
- Параметр NOTIFICATION поддерживает новые значения Notify Message Type.
- Поле алгоритма HIP_SIGNATURE увеличено в размере с 8 битов до 16 для выравнивания с HOST_ID.
- Спецификация уточняет, что параметр SEQ всегда содержит 1 идентификатор обновления, но ACK может подтверждать несколько идентификаторов обновлений.
- Удалено ограничение, требовавшее наличие лишь одного параметра ECHO_RESPONSE_UNSIGNED в каждом пакете HIP.
- Документ создаёт новый диапазон для выделения параметров, охватываемых подписью (при её наличии) и применяет его к новому параметру DH_GROUP_LIST.
- Документ разъясняет, что в пакете может быть несколько параметров NOTIFY.

Ниже перечислены изменения в содержимом пакетов (параграф 5.3).

- Пакет I1 сейчас включает DH_GROUP_LIST Инициатора.
- Пакет R1 сейчас включает параметры HIP_CIPHER, HIT_SUITE_LIST, DH_GROUP_LIST, TRANSPORT_FORMAT_LIST.
- Пакет I2 сейчас включает параметры HIP_CIPHER и TRANSPORT_FORMAT_LIST.
- Документ разъясняет, что пакеты UPDATE, не включающие параметр SEQ или ACK, недействительны.

11. Литература

11.1. Нормативные документы

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

[NIST.800-131A.2011] National Institute of Standards and Technology, "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths", NIST SP 800-131A, January 2011, <http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2119] 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>.

[RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998, <http://www.rfc-editor.org/info/rfc2404>.

[RFC2410] Glenn, R. and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998, <http://www.rfc-editor.org/info/rfc2410>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2536] Eastlake 3rd, D., "DSA KEYs and SIGs in the Domain Name System (DNS)", RFC 2536, March 1999, <http://www.rfc-editor.org/info/rfc2536>.

[RFC3110] Eastlake 3rd, D., "RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)", RFC 3110, May 2001, <http://www.rfc-editor.org/info/rfc3110>.

[RFC3526] 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>.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003, <http://www.rfc-editor.org/info/rfc3602>.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005, <http://www.rfc-editor.org/info/rfc3972>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005, <http://www.rfc-editor.org/info/rfc4034>.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005, <http://www.rfc-editor.org/info/rfc4282>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC4754] Fu, D. and J. Solinas, "IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 4754, January 2007, <http://www.rfc-editor.org/info/rfc4754>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, May 2007, <http://www.rfc-editor.org/info/rfc4868>.

[RFC5702] Jansen, J., "Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC", RFC 5702, October 2009, <http://www.rfc-editor.org/info/rfc5702>.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC7343] Laganier, J. and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers Version 2 (ORCHIDv2)", RFC 7343, September 2014, <http://www.rfc-editor.org/info/rfc7343>.

[RFC7402] Jokela, P., Moskowitz, R., and J. Melen, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 7402, April 2015, <http://www.rfc-editor.org/info/rfc7402>.

11.2. Дополнительная литература

[AUR05] Aura, T., Nagarajan, A., and A. Gurtov, "Analysis of the HIP Base Exchange Protocol", in Proceedings of the 10th Australasian Conference on Information Security and Privacy, July 2005.

[CRO03] Crosby, S. and D. Wallach, "Denial of Service via Algorithmic Complexity Attacks", in Proceedings of the 12th USENIX Security Symposium, Washington, D.C., August 2003.

[DIF76] Diffie, W. and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory Volume IT-22, Number 6, pages 644-654, November 1976.

[FIPS.186-4.2013] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS PUB 186-4, July 2013, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[FIPS.197.2001] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[HIP-ARCH] Moskowitz, R., Ed., and M. Komu, "Host Identity Protocol Architecture", Work in Progress[1], draft-ietf-hip-rfc4423-bis-09, October 2014.

[HIP-DNS-EXT] Laganier, J., "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", Work in Progress[2], draft-ietf-hip-rfc5205-bis-06, January 2015.

[HIP-HOST-MOB] Henderson, T., Ed., Vogt, C., and J. Arkko, "Host Mobility with the Host Identity Protocol", Work in Progress[3], draft-ietf-hip-rfc5206-bis-08, January 2015.

[HIP-REND-EXT] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", Work in Progress[4], draft-ietf-hip-rfc5204-bis-05, December 2014.

[KAU03] Kaufman, C., Perlman, R., and B. Sommerfeld, "DoS protection for UDP-based protocols", in Proceedings of the 10th ACM Conference on Computer and Communications Security, October 2003.

[KRA03] Krawczyk, H., "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols", in Proceedings of CRYPTO 2003, pages 400-425, August 2003.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981, <http://www.rfc-editor.org/info/rfc792>.

[RFC2785] Zuccherato, R., "Methods for Avoiding the "Small-Subgroup" Attacks on the Diffie-Hellman Key Agreement Method for S/MIME", RFC 2785, March 2000, <http://www.rfc-editor.org/info/rfc2785>.

[RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, September 2000, <http://www.rfc-editor.org/info/rfc2898>.

[RFC3447] 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>.

[RFC3849] Huston, G., Lord, A., and P. Smith, "IPv6 Address Prefix Reserved for Documentation", RFC 3849, July 2004, <http://www.rfc-editor.org/info/rfc3849>.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008, <http://www.rfc-editor.org/info/rfc5201>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[1] Работа опубликована в RFC 9063. *Прим. перев.*
[2] Работа опубликована в RFC 8005. *Прим. перев.*
[3] Работа опубликована в RFC 8046. *Прим. перев.*
[4] Работа опубликована в RFC 8004. *Прим. перев.*

[RFC5338] Henderson, T., Nikander, P., and M. Komu, "Using the Host Identity Protocol with Legacy Applications", RFC 5338, September 2008, <http://www.rfc-editor.org/info/rfc5338>.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009, <http://www.rfc-editor.org/info/rfc5533>.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, January 2010, <http://www.rfc-editor.org/info/rfc5737>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010, <http://www.rfc-editor.org/info/rfc5869>.

[RFC5903] Fu, D. and J. Solinas, "Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2", RFC 5903, June 2010, <http://www.rfc-editor.org/info/rfc5903>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2011, <http://www.rfc-editor.org/info/rfc6090>.

[RFC6253] Heer, T. and S. Varjonen, "Host Identity Protocol Certificates", RFC 6253, May 2011, <http://www.rfc-editor.org/info/rfc6253>.

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, December 2013, <http://www.rfc-editor.org/info/rfc7045>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[RSA] Rivest, R., Shamir, A., and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM 21 (2), pp. 120-126, February 1978.

[SECG] SECG, "Recommended Elliptic Curve Domain Parameters", SEC 2 Version 2.0, January 2010, <http://www.secg.org/>.

Приложение A. Использование головоломок Ответчиком

Как указано в параграфе 4.1.1, Ответчик может задержать создание состояния и отвергнуть большинство фиктивных пакетов I2, используя набор заранее созданных пакетов R1 и локальную функцию выбора. В этом приложении описан один из вариантов реализации механизма, чтобы дать разработчикам идею. При использовании этого приложения разработчики могут внести изменения, усложняющие злоумышленникам организацию атаки.

Ответчик создает секретное значение S, которое затем обновляет периодически. При этом Ответчик должен помнить 2 последних значения S. При каждой регенерации S счётчик генерации (R1_COUNTER) R1 увеличивается на 1.

Ответчик создаёт подписанный заранее пакет R1. Подписи для созданных заранее пакетов R1 должны пересчитываться при повторном расчёте ключа DH или смене значения R1_COUNTER при регенерации S.

Когда Инициатор передаёт пакет I1 для инициализации соединения, Ответчик получает из пакета тег HIT и адрес IP и генерирует значение #I для головоломки. Значение #I устанавливается для заранее подписанного пакета R1.

```
#I = Ltrunc( RHASH ( S | HIT-I | HIT-R | IP-I | IP-R ), n)
```

где n = RHASH_len. Алгоритм RHASH совпадает с применяемым при генерации тега HIT для Ответчика.

Во входящем пакете I2 Ответчик получает сведения, требуемые для проверки решения головоломки: теги HIT, адреса IP и сведения об используемом значение S из R1_COUNTER. По этим значениям Ответчик может снова создать значение #I и сравнить его с полученным в пакете I2. Если значения #I совпадают, он может проверить решение головоломки, используя #I, #J и #K (сложность), в противном случае пакет I2 отбрасывается.

```
V := Ltrunc( RHASH( I2.I | I2.hit_i | I2.hit_r | I2.J ), #K )
если V != 0, пакет отбрасывается
```

Если решение головоломки верное, значения #I и #J сохраняются для последующего применения в качестве входных данных при генерации ключевого материала.

Сохранение сведений об ошибке в решении головоломки зависит от реализации. Хотя Ответчик не обязан сохранять данные состояния, он может делать это для защиты себя от некоторых атак (см. параграф 4.1.1).

Приложение B. Генерация кодирования открытого ключа из HI

Приведённый ниже псевдокод иллюстрирует процесс генерации ключа шифрования из HI для RSA и DSA. Символ := указывает назначение (присваивание), += - добавление в конец. Псевдофункция encode_in_network_byte_order принимает два параметра - целое число (bignum) и размер в байтах, возвращая целое число в форме строки байтов заданного размера.

```
switch ( HI.algorithm )
{

case RSA:
   buffer := encode_in_network_byte_order ( HI.RSA.e_len,
             ( HI.RSA.e_len > 255 ) ? 3 : 1 )
   buffer += encode_in_network_byte_order ( HI.RSA.e, HI.RSA.e_len )
   buffer += encode_in_network_byte_order ( HI.RSA.n, HI.RSA.n_len )

   break;

case DSA:
   buffer := encode_in_network_byte_order ( HI.DSA.T , 1 )
   buffer += encode_in_network_byte_order ( HI.DSA.Q , 20 )
   buffer += encode_in_network_byte_order ( HI.DSA.P , 64 +
```

```
                                        8 * HI.DSA.T )
    buffer += encode_in_network_byte_order ( HI.DSA.G , 64 +
                                        8 * HI.DSA.T )
    buffer += encode_in_network_byte_order ( HI.DSA.Y , 64 +
                                        8 * HI.DSA.T )

    break;

}
```

Приложение C. Примеры контрольных сумм для пакетов HIP

Контрольная сумма HIP для пакетов HIP задана в параграфе 5.1.1, контрольные суммы пакетов TCP и UDP, передаваемых через защищённые связи с поддержкой HIP, - в параграфе 4.5.1. Приведённые ниже примеры используют адреса [RFC3849] и [RFC5737], а также HIT с префиксом 2001:20, за которым следуют нули, а затем десятичное значение 1 и 2, соответственно.

Приведённый ниже пример предназначен лишь для тестирования расчёта контрольной суммы.

C.1. IPv6 HIP (пакет I1)

Адрес отправителя:	2001:db8::1	
Адрес получателя:	2001:db8::2	
Размер пакета вышележ. уровня:	48	0x30
Следующий заголовок:	139	0x8b
Протокол Payload:	59	0x3b
Размер заголовка:	5	0x5
Тип пакета:	1	0x1
Версия:	2	0x2
Резерв:	1	0x1
Управление:	0	0x0
Контрольная сумма:	6750	0x1a5e
HIT отправителя:	2001:20::1	
HIT получателя:	2001:20::2	
Тип DH_GROUP_LIST:	511	0x1ff
Размер DH_GROUP_LIST:	3	0x3
DH_GROUP_LIST Group ID :	3,4,8	

C.2. IPv4 HIP (пакет I1)

Адрес отправителя:	192.0.2.1	
Адрес получателя:	192.0.2.2	
Размер пакета вышележ. уровня:	48	0x30
Следующий заголовок:	139	0x8b
Протокол Payload:	59	0x3b
Размер заголовка:	5	0x5
Тип пакета:	1	0x1
Версия:	2	0x2
Резерв:	1	0x1
Управление:	0	0x0
Контрольная сумма:	61902	0xf1ce
HIT отправителя:	2001:20::1	
HIT получателя:	2001:20::2	
Тип DH_GROUP_LIST:	511	0x1ff
Размер DH_GROUP_LIST:	3	0x3
DH_GROUP_LIST Group ID:	3,4,8	

C.3. Сегмент TCP

Независимо от протокола IPv6 или IPv4 сокеты TCP и UDP используют формат псевдозаголовка IPv6 [RFC2460] с HIT вместо адресов IPv6.

HIT отправителя:	2001:20::1	
Receiver's HIT:	2001:20::2	
Размер пакета вышележ. уровня:	20	0x14
Следующий заголовок:	6	0x06
Порт отправителя:	65500	0xffdc
Порт получателя:	22	0x0016
Порядковый номер:	1	0x00000001
Номер подтверждения:	0	0x00000000
Смещение данных:	5	0x5
Флаги:	SYN	0x02
Размер окна:	65535	0xffff
Контрольная сумма:	28586	0x6faa
Указатель важности:	0	0x0000

Приложение D. 160-битовые группы ECDH и ECDSA

160-битовые группы ECDH и ECDSA SECP160R1 рассчитаны на симметричную стойкость 80 битов. Когда-то это считалось достаточным для защиты в течение года. Сегодня эти группы следует применять лишь для хостов с невысокой производительностью (например, встраиваемые устройства) и при низких требованиях к безопасности (например, не нужна долгосрочная защита конфиденциальности).

Приложение E. HIT Suite и генерация HIT

HIT как идентификатор ORCHID [RFC7343] состоит из трёх частей - 28-битовый префикс, 4-битовый код алгоритма генерации ORCHID (OGA) и хэш-значения, включающее Host Identity и идентификатор контекста. Индекс OGA указывает конкретный алгоритм, с помощью которого создаётся открытый ключ и 96-битовое хэшированное

кодирование. OGA зависит от протокола и интерпретируется, как описано ниже, для всех протоколов, использующих общий с HIP идентификатор контекста. Группы HIP задают действительные комбинации алгоритмов подписи и хэширования в HIT Suite. Эти наборы HIT Suite указываются индексом, который передаётся в поле OGA ID идентификаторов ORCHID.

Набор применяемых HIT Suite будет расширяться по мере роста вычислительных возможностей и обнаружения уязвимостей в применяемых алгоритмах. Предполагается использование HIT Suite для внедрения новых наборов и отказа от старых, пока они ещё не стади опасными. Поскольку 4-битовое поле OGA ID позволяет иметь лишь 15 HIT Suite одновременно (HIT Suite ID = 0 является резервным), идентификаторы отмененных HIT Suite потребуется применять снова. В таких случаях может просходить «опрокидывание» (rollover) HIT Suite ID и недавно добавленный HIT Suite будет иметь меньший индекс, нежели его предшественник. «Опрокидование» фактически препятствует повторному использованию HIT Suite. Для плавного перехода следует отменять HIT Suite заблаговременно до повторного использования индекса HIT Suite.

Поскольку число HIT Suite строго ограничено значением 16, выбирать HIT Suite следует с осторожностью, поэтому в HIT Suite группируются подходящие наборы алгоритмов.

HIT Suite в HIT Ответчика определяет RHASH и хэш-функцию, которая будет применяться для HMAC в пакетах HIP, а также семейство алгоритмов подписи для генерации HI. Список HIT Suite приведён в таблице 10.

Благодарности

Стремление создать протокол HIP возникло после заседания MALLOC на 43-й конференции IETF. Baiju Patel и Hilarie Orman реально помогли исходному автору Bob Moskowitz вывести HIP за рамки 5 абзацев идей. Протокол обрел зрелость благодаря значительному вкладу членов IETF. Самое главное, что цели разработки были чётко сформулированы и отличались от других работ в этом направлении. Особо следует отметить членов исследовательской группы IRTF NameSpace. Noel Chiappa внёс ценный влад на ранних этапах обсуждения обработки идентификаторов, а Keith Moore - толчок для обеспечения распознаваемости. Steve Deering призывал продолжать работу, поскольку стойкость могла служить подтверждением перспективности идей для исследовательской группы.

Много людей внесло вклад в работу - ценные советы по безопасности дал Steve Bellovin, Rob Austein следил за связанными с DNS частями, Call Kocher научил Bob Moskowitz, как сделать головоломки сложными для ответа Инициатору, но простыми для проверки Ответчиком. Bill Sommerfeld предложил концепцию днейрождения, которая позднее превратилась в счётчик генерации R1, для простого управления перезагрузкой. Erik Nordmark предложил механизм CLOSE для завершения соединений, Rodney Thayer и Hugh Daniels предоставили множество откликов. На ранних стадиях разработки документа John Gilmore заставлял Bob Moskowitz предлагать что-то важное.

На более поздних этапах работы, когда редактирование было передано Pekka Nikander, вклад первых разработчиков оказался неоценимым. Без реальных реализаций протокола документ не достиг бы требуемого уровня.

Как обычно бывает в IETF, много людей внесло свой вклад в текст и идеи. Список этих людей включает Jeff Ahrenholz, Francis Dupont, Derek Fawcus, George Gross, Xin Gu, Rene Hummen, Miika Komu, Mika Kousa, Julien Laganier, Andrew McGregor, Jan Melen, Henrik Petander, Michael Richardson, Tim Shepard, Jorma Wall, Jukka Ylitalo. Приносим свои извинения тем, кого забыли включить в этот список.

После создания в начале 2004 г. рабочей группы HIP в документ было внесено множество изменений. В частности, исходный документ был поделен на две части, одна из которых описывала базовый обмен, а другая - использование ESP. Некоторые изменения протокола, предлоенные Aura и др. [AUR05], были добавлены на финальном этапе.

Адреса авторов

Robert Moskowitz (редактор)
HTT Consulting
Oak Park, MI
United States
EMail: rgm@labs.htt-consult.com

Tobias Heer
Hirschmann Automation and Control
Stuttgarter Strasse 45-51
Neckartenzlingen 72654
Germany
EMail: tobias.heer@belden.com

Petri Jokela
Ericsson Research NomadicLab
Jorvas FIN-02420
Finland
Phone: +358 9 299 1
EMail: petri.jokela@nomadiclab.com

Thomas R. Henderson
University of Washington
Campus Box 352500
Seattle, WA
United States
EMail: tomhend@u.washington.edu

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru