Internet Engineering Task Force (IETF)
Request for Comments: 9258
Category: Standards Track
ISSN: 2070-1721

D. Benjamin
Google, LLC.
C. A. Wood
Cloudflare

July 2022

Importing External Pre-Shared Keys (PSKs) for TLS 1.3

Импорт внешних ключей PSK для TLS 1.3

Аннотация

Этот документ описывает интерфейс для импорта внешних заранее распространённых ключей (Pre-Shared Key или PSK) в TLS 1.3.

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 7841.

Информация о текущем статусе документа, найденных ошибках и способах обратной связи доступна по ссылке https://www.rfc-editor.org/info/rfc9258.

Оглавление

1. Введение

TLS 1.3 [RFC8446] поддерживает аутентификацию с заранее распределенным ключом (PSK), при этом ключи PSK могут устанавливаться с помощью сеансовых квитанций (ticket) из предшествующих соединений или с помощью специального (out-of-band) механизма. Протокол требует, чтобы каждый ключ PSK использовался лишь с одной хэш-функцией. Это сделано для упрощения анализа протокола. В TLS 1.2 [RFC5246] такого требования нет и PSK может применяться с любым алгоритмом хэширования и псевдослучайной функцией (pseudorandom function или PRF) TLS 1.2. Хотя не известно способа, каким один и тот же внешний PSK может давать связанный вывод в TLS 1.3 и предшествующих версиях, был проведён лишь ограниченный анализ. Приложениям **следует** предоставлять раздельные PSK для (O)TLS 1.3 и прежних версий. Если это невозможно (например, внешние PSK уже развёрнуты или имеются иные ограничения), повторное использование PSK в разных версиях TLS может давать связанный вывод, который, в свою очередь, ведёт к проблемам безопасности (см. Приложение E.7 к [RFC8446]).

Для смягчения проблем этот документ задаёт интерфейс импортёра PSK. С помощью которого можно импортировать внешние PSK и потом привязать их к конкретной функции вывода ключей (key derivation function или KDF) и хэш-функции для использования в TLS 1.3 [RFC8446] и DTLS 1.3 [RFC9147]. В частности, описывается механизм для импорта PSK, выведенных из внешних PSK путём включения целевой KDF, версии протокола (D)TLS и необязательной

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

строки контекста для обеспечения уникальности. Этот процесс даёт набор PSK-кандидатов, каждый из которых привязан к целевой KDF и протоколу, которые отличаются от применяемых в (D)TLS 1.2 и прежних версий. Это преобразует отдельный PSK и отождествление в набор PSK и отождествлений.

2. Принятые соглашения

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе должны интерпретироваться в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они выделены шрифтом, как показано здесь.

3. Терминология

Ниже приведены определения используемых в документе терминов.

External PSK (EPSK) - внешний PSK
: Ключ PSK организованный или предоставленный внешними средствами (out of band), т. е не в соединении TLS, который представляет собой кортеж (Base Key, External Identity, Hash).

Base Key - базовый ключ
: Секретное значение EPSK.

External Identity - внешнее отождествление (свидетельство)
: Последовательность байтов, служащая для идентификации EPSK.

Target protocol - целевой протокол
: Протокол, для которого импортируется PSK.

Target KDF - целевая KDF
: Функция KDF, для которой импортируется PSK.

Imported PSK (IPSK) - импортированный PSK
: Ключ TLS PSK выведенный из EPSK, необязательной строки контекста, целевого протокола и целевой KDF.

Non-imported PSK - неимпортированный PSK
: Ключ EPSK, испольуемый в качестве TLS PSK напрямую (без импорта).

Imported Identity - импортированное отождествление
: Последовательность байтов, служащая для идентификации IPSK.

В документе применяется язык представления из раздела 3 в [RFC8446].

4. Обзор

Интерфейс импортёра PSK отражает интерфейс экспортёра TLS (см. [RFC8446]) в том смысле, что он меняет ключ на основе некоторой контекстной информации. В отличие от интерфейса экспортёра, где уникальность вывода достигается за счёт явной метки и строки контекста, определённый здесь интерфейс импортёра принимает внешний ключ PSK и отождествление, «импортируя» их в TLS путём создания набора «производных» PSK и отождествлений, которые уникальны. Каждый из этих выведенных PSK привязывается к целевому протоколу, идентификатору KDF и необязательной строке контекста. Кроме того, полученные ключи изменяются с использованием новой метки вывода для предотвращения конфликтов с неимпортированными PSK. Через этот интерфейс импорт внешних PSK с разными отждествлениями даёт разные ключи привязки PSK (binder).

Использование импортированных ключей не требуется согласовывать, поскольку клиент и сервер не согласуют отождествления, если импорт был некорректен. Конечные точки могут постепенно развёртывать поддержку импортёров PSK, предлагая неимпортированные PSK для TLS версий до TLS 1.3. Неимпортированные и импортированные PSK не эквивалентны, поскольку из отождествления различаются (см. раздел 7).

Конечным точкам, импортирующим внешние ключи, **недопустимо** применять служащие вводом в процесс импорта ключи для каких-либо иных целей и **недопустимо** применять выведенные ключи для целей, отличных от TLS PSK. Кроме того, каждый внешний PSK, введенный в процесс импортёра, должен быть связан не более чем с одной хэш-функцией. Это аналогично правилам параграфа 4.2.11 [RFC8446]. Дополнительное обсуждение приведено в разделе 8.

5. Импортёр PSK

В этом разделе описан интерфейс импортёра PSK и лежащий в его основе механизм диверсификации, а также изменение расчёта связующего ключа (binder key).

5.1. Диверсификация внешнего PSK

На входе интерфейс импортёра PSK принимает EPSK с внешним отождествлением (External Identity) external_identity и базовым ключом epsk (определены в разделе 3) вместе с необязательным контекстом. Затем ввод преобразуется в набор PSK и импортированных отождествлений для использования в соединении с целевым протоколом и KDF. В частности, для каждого поддерживаемого целевого протокола target_protocol и KDF target_kdf импортёр создаёт структуру ImportedIdentity, показанную ниже.

```
struct {
   opaque external_identity<1...2^16-1>;
   opaque context<0..2^16-1>;
   uint16 target_protocol;
   uint16 target_kdf;
} ImportedIdentity;
```

Список значений ImportedIdentity.target_kdf поддерживается IANA, как указано в разделе 10. Внешние PSK **недопустимо** импортировать для (D)TLS 1.2 и более ранних версий. В разделе 7 обсыждается сосуществование импортированных PSK для TLS 1.3 с неимпортированными PSK более ранних версий при поэтапном развертывании.

Значение ImportedIdentity.context **должно** включать контекст, используемый для определения EPSK, если тот имеется. Например, ImportedIdentity.context может включать сведения о ролях партнёров или отождествления для смягчения

атак с отражением в стиле Selfie (Selfie-style reflection attack) [Selfie] (см. Приложение A). Поскольку EPSK является ключом, выведенным из внешнего протокола или последовательности протоколов, значение ImportedIdentity.context **должно** включать привязку каналов для производных протоколов [RFC5056]. Детали этой привязки зависят от протокола и выходят за рамки этой спецификации.

Значение ImportedIdentity.target_protocol **должно** указывать версию протокола (D)TLS, для которой импортируется PSK. Например, TLS 1.3 [RFC8446] использует значение 0x0304, которое применяется также в QUICv1 [QUIC]. Отметим, что это означает зависимость числа PSK, выводимых из EPSK, от числа целевых протоколов.

На основе ImportedIdentity и соответствующего EPSK с базовым ключом epsk, импортированный PSK IPSK с базовым ключом ipskx вычисляется, как показано ниже.

```
epskx = HKDF-Extract(0, epsk)
ipskx = HKDF-Expand-Label(epskx, "derived psk", Hash(ImportedIdentity), L)
```

L соответствует размеру вывода KDF ImportedIdentity.target_kdf, как указано в разделе 10. Для основанных на хэшировании KDF, таких как HKDF_SHA256 (0x0001), это будет размер результата хэш-функции, например, 32 октета для SHA256. Ключ IPSK должен иметь размер, подхожящий для поддерживаемых шифров. Внутри HKDF-Expand-Label применяет метку, соответствующую ImportedIdentity.target_protocol (например, "tls13" для TLS 1.3 в соответствии с параграфом 7.1 [RFC8446] или "dtls13" для DTLS 1.3 в соответствии с параграфом Section 5.10 [RFC9147]).

Отождествлением ipskx при передаче в линию служит ImportedIdentity, т. е. преобразованное в последовательную форму содержимое ImportedIdentity выступает как содержимое PskIdentity.identity в расширении PSK. Соответствующим вводом PSK для планирования ключей TLS 1.3 является "ipskx".

Поскольку максимальный размер расширения PSK составляет 2^{16} - 1 октетов, импортированное отождествление (Imported Identity), превышающее этот размер, может вызывать ошибку декодирования. Поэтому интерфейсу импортёра PSK **следует** отвергать ImportedIdentity с размером, превышающим это значение.

Хэш-функция, используемая для функции вывода ключей на основе HMAC (HMAC-based Key Derivation Function или HKDF) [RFC5869], связана с EPSK. Это не хэш-функция, связанная с ImportedIdentity.target_kdf. Если EPSK не имеет связанной хэш-функции, **следует** применять SHA-256 [SHA2]. Диверсификация EPSK с помощью ImportedIdentity.target_kdf гарантирует, что IPSK применяется в качестве входного ключевого материала KDF не более 1 раза, что соответствует требованиям [RFC8446] (см. раздел 8).

Конечным точкам **следует** генерировать совместимые отождествления ipskx для каждого целевого шифра, который точка поддерживает. Например, импорт ключей для TLS_AES_128_GCM_SHA256 и TLS_AES_256_GCM_SHA384 будет давать два PSK, один для HKDF-SHA256, другой для HKDF-SHA384. При поддержке TLS_AES_128_GCM_SHA256 и TLS_CHACHA20_POLY1305_SHA256, напротив, нужен лишь 1 производный ключ. Каждый шифр однозначно указывает целевую функцию KDF. Будущие спецификации, меняющие способ согласования KDF, должны будут обновить эту спецификацию для разъяснения выбора целевых KDF в процессе импорта.

EPSK **можно** импортировать до начала соединения, если целевые KDF, протоколы и строки контекста известны заранее. **Можно** также импортировать EPSK для упреждающего использования данных, если они привязаны к настройкам протокола и конфигурации, которые требуются для упреждающей передачи данных. По минимуму это означает, что вместе с EPSK **должны** быть предоставлены значение согласования протокола прикладного уровня (Application-Layer Protocol Negotiation или ALPN) [RFC7301], транспортные параметры QUIC (при использовании с QUIC) и другие связанные параметры, согласуемые для упреждающих данных.

5.2. Вывод связующего ключа

Для предотвращения путаницы между импортированными и неимпортированными PSK в импортированных PSK меняется метка вывода связующего ключа PSK. В частности, вычисление стандартного связующего ключа TLS 1.3 PSK выполняется, как показано ниже.

```
            0
            |
            v
PSK ->  HKDF-Extract = Early Secret
            |
            +-----> Derive-Secret(., "ext binder" | "res binder", "")
            |                     = binder_key
            v
```

Импортированные PSK при выводе binder_key используют строку "imp binder" вместо "ext binder" или "res binder" и расчёт связующего ключа имеет вид

```
            0
            |
            v
PSK ->  HKDF-Extract = Early Secret
            |
            +-----> Derive-Secret(., "ext binder"
            |                      | "res binder"
            |                      | "imp binder", "")
            |                     = binder_key
            v
```

Эта новая метка гарантирует, что клиент и сервер будут согласовывать применение внешнего PSK тогда и только тогда, когда (a) обе конечные точки импортируют PSK или (b) ни одна из них не делает этого. Поскольку binder_key является ключом листа, изменение расчёта не влияет на другие ключи.

6. Отмена хэш-функций

Клиент или сервер, желающий отменить хэш-функцию и больше не применять её для TLS 1.3, удаляет соответствующую KDF из числа целевых KDF при импорет ключей. Это не влияет на работу KDF, служащих для вывода импортируемых PSK.

7. Постепенное развёртывание

В системах, где PSK уже подготовлены для применения с TLS 1.2, попытка поэтапного развёртывания механизма импорта приведёт к одновременному использованию уже предоставленных PSK напрямую как TLS 1.2 PSK и как EPSK, что, в свою очередь, означает возможность применения одних KDF и ключа а двух разных контекстах протокола. Это не рекомендуется (см. раздел 8). Однако преимущества применения TLS 1.3 и импортёров PSK могут быть достаточно убедительными для выбора в существующем развёртывании такой несовместимой конфигурации на короткий период развёртывания новых программ (использующих TLS 1.3 и импортёры). Операторам рекомендуется делать этот период как можно короче.

8. Вопросы безопасности

Цель безопасности импортёра PSK можно грубо сформулировать как предотвращение повторного применения PSK в KDF при надлежащей аутентификации конечных точек. При моделировании в качестве вычислительных экстракторов KDF предполагают, что входной ключевой материал (input keying material или IKM) отбирается из некого «источника» распределения вероятности и любые два значения IKM выбирабтся независимо одно от другого [Kraw10]. Это требование независимости от источника предполагает, что одно значение IKM не может применяться в разных KDF.

Аутентификация на базе PSK функционально эквивалентна возобновлению сессии, поскольку соединение применяет имеющийся ключевой материал для аутентификации обеих конечных точек. В соответствии с [BAA15] это является формой композитной аутентификации. Такая аутентификация, грубо говоря, является свойством, состоящим в том, что выполнение нескольких протоколов аутентификации, из которых хотя бы 1 не скомпрометирован, совместно аутентифицирует все протоколы. Поэтому аутентификации с предоставленными извне PSK в идеале следует аутентифицировать как соединение TLS, так и внешний процесс представления. Обычно этот процесс создаёт PSK и соответствующий контекст, где PSK был выведен и где его следует применять. При доступности контекст используется как значение ImportedIdentity.context. Внешний PSK без такого контекста называется безконтекстным (context-free).

Таким образом, с учётом требований независимости от источника и составной аутентификации описанный в документе интерфейс импортёра PSK направлен на достижение указанных ниже целей.

1. Импорт предоставленных извне PSK в соединение TLS обеспечивает составную аутентификацию соединения и процесса представления.
2. Безконтекстные PSK обеспечивают аутентификацию лишь в контексте одного соединения.
3. Импортированные PSK не применяются в качестве IKM для двух разных KDF.
4. Импортированные PSK не будут конфиликтовать с будущими версиями протокола и функций KDF.

Нет каких-либо известных данных или проблем безопасности, вызываемых процессом расчёта импортируемых PSK из внешниз PSK и обработки имеющихся внешних PSK, применяемых в (D)TLS 1.2 и ниже, как указано в разделе 7. Однако был проведён лишь ограниченный анализ и это является ещё одной причиной, почему приложениям следует предоставлять отдельные PSK для (D)TLS 1.3 и прежних версий даже с интерфейсом импортёра (D)TLS 1.3.

Импортёр PSK не предотвращает создание приложениями отождествлений PSK без импорта, конфликтующих с импортированными отождествлениями PSK.

9. Вопросы приватности

Внешние отождествления PSK обычно имеют статическую структуру, поэтому конечные точки могут применять их для поиска ключевого материала. В некоторых системах и вариантах применения они могут служить для отслеживания.

Отметим, что ImportedIdentity.context является открытым текстом в линии как часть отождествления PSK. Без защиты механизмами вроде TLS Encrypted ClientHello [ECH] приложениям **не следует** помещать в это поле приватные данные.

10. Взаимодействие с IANA

Агентство IANA создало реестр TLS KDF Identifiers в имеющемся реестре Transport Layer Security (TLS) Parameters. Записи реестра показаны в таблице 1

Таблица 1. Реестр идентификаторов TLS KDF.

Значение	***Описание KDF***	***Документ***
0x0000	Резерв	RFC 9258
0x0001	HKDF_SHA256	[RFC5869]
0x0002	HKDF_SHA384	[RFC5869]

Новые значения для целевых KDF выделяются в соответствии с указанными ниже процедурами:

- для диапазона 0x0000-0xfeff применяется процедура Specification Required [RFC8126];
- значения 0xff00-0xffff зарезервированы для приватного использования (Private Use) [RFC8126].

Процедуры регистрации значений в пространстве Specification Required заданы в разделе 17 [RFC8447].

11. Литература

11.1. Нормативные документы

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <https://www.rfc-editor.org/info/rfc5056>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/info/rfc8447>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

11.2. Дополнительная литература

[BAA15] Bhargavan, K., Delignat-Lavaud, A., and A. Pironti, "Verified Contributive Channel Bindings for Compound Authentication", Proceedings 2015 Network and Distributed System Security, DOI 10.14722/ndss.2015.23277, February 2015, <https://doi.org/10.14722/ndss.2015.23277>.

[ECH] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-14, 13 February 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-14>.

[Kraw10] Krawczyk, H., "Cryptographic Extraction and Key Derivation: The HKDF Scheme", Proceedings of Crypto 2010, May 2010, <https://eprint.iacr.org/2010/264>.

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[Selfie] Drucker, N. and S. Gueron, "Selfie: reflections on TLS 1.3 with PSK", DOI 10.1007/s00145-021-09387-y, May 2021, <https://eprint.iacr.org/2019/347.pdf>.

[SHA2] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

Приложение A. Атаки Selfie

Атаки Selfie [Selfie] основаны на злоупотреблении интерфейсом PSK, неявно предполагающим, что каждый ключ PSK известен лишь клиенту и серверу. Если несколько клиентов или серверов с разными ролями используют общий ключ PSK, TLS аутентифицирует лишь группу целиком. Узел аутентифицирует своего партнёра из группы независимо от подлинности того. Отметим, что такое возможно и в случае совместного применения PSK двумя узлами без ролей.

Приложения, требующие аутентификации по ролям при использовании общего PSK для всех узлов, могут решить проблему путём передачи сведений о ролях через соединение TLS после согласования или путём включения роли клиента и сервера в строку контекста IPSK. Например, если приложение идентифицирует каждый узел по MAC-адресу (Media Access Control), оно может использовать приведённую ниже строку контекста.

```
struct {
  opaque client_mac<0..2^8-1>;
  opaque server_mac<0..2^8-1>;
} Context;
```

Если злоумышленник затем переправит сообщение ClientHello, предназначенное одному узлу, на другой узел, включая создателя ClientHello, получатель определит другую строку контекста и согласование не завершится успехом.

Отметим, что в этом варианте по=прежнему применяется один общий ключ PSK для всех узлов поэтому каждый узел должен быть доверенным, чтобы он не взял на себя чужую роль (прикинулся другим узлом).

Благодарности

Авторы признательны Eric Rescorla и Martin Thomson за дискуссии, которые привели к созданию этого документа, а также Christian Huitema за вклад в соображения по приватности внешних PSK. John Preuß Mattsson предоставил сведения по части развёртывания импортёров PSK. Hugo Krawczyk предоставил рекомендации по безопасности. Martin Thomson, Jonathan Hoyland, Scott Hollenbeck, Benjamin Kaduk и другие предоставили рецензии, отклики и предложения по улучшению документа.

Адреса авторов

David Benjamin
Google, LLC.
Email: davidben@google.com

Christopher A. Wood
Cloudflare
Email: caw@heapingbits.net

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru