Internet Engineering Task Force (IETF)
Request for Comments: 9580
Obsoletes: 4880, 5581, 6637
Category: Standards Track
ISSN: 2070-1721

P. Wouters, Ed.
Aiven
D. Huigens
Proton AG
J. Winter
Sequoia PGP
Y. Niibe
FSIJ
July 2024

OpenPGP

Аннотация

Этот документ задаёт формат сообщений, применяемых в OpenPGP. OpenPGP обеспечивает шифрование с использованием асимметричных (с открытым ключом) и симметричных алгоритмов, цифровые подписи, сжатие и управление ключами.

Документ создан с целью публикации всех сведений, требуемых для разработки функционально совместимых приложений на основе формата OpenPGP. Это не справочник для пошаговой разработки приложений. Описываются лишь формат и методы, требуемые для чтения, проверки, генерации и записи соответствующих пакетов, проходящих через любую сеть. Методы хранения и реализации не рассматриваются, однако обсуждаются вопросы реализации, связанные с предотвращением изъянов защиты.

Этот документ отменяет RFC 4880 (OpenPGP Message Format), 5581 (The Camellia Cipher in OpenPGP), 6637 (Elliptic Curve Cryptography (ECC) in OpenPGP).

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.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc9580.

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 содержит сведения о форматах пакетов обмена сообщениями, используемых OpenPGP для шифрования и дешифрования, подписей и управления ключами. Это пересмотр [RFC4880] (OpenPGP Message Format), который пересматривает [RFC2440], заменивший [RFC1991] (PGP Message Exchange Formats).

Документ отменяет [RFC4880] (OpenPGP), [RFC5581] (Camellia in OpenPGP), [RFC6637] (Elliptic Curves in OpenPGP). На момент создания в документе учтены все подтверждённые сведения об ошибках, которые указаны в Приложении C.

Для программ, в которых уже реализованы предыдущие спецификации, может быть полезен обзор из Приложения B, где указаны основные изменения.

1.1. Уровни требований

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

Ключевые слова Private Use (приватное применение), Specification Required (требуется спецификация), RFC Required (требуется RFC) в этом документе служат для описания процедур выделения пространств имён, как описано в [RFC8126].

Некоторые из используемых в документе технологий были улучшены с момента выпуска предыдущей спецификации OpenPGP. Подробности приведены в Приложении B.1.

2. Базовые функции

OpenPGP обеспечивает конфиденциальность и целостность данных для сообщений и файлов за счёт использования шифрования с открытым и/или симметричным ключом и цифровых подписей. Обеспечиваются форматы для кодирования и передачи зашифрованных и/или подписанных сообщений. Кроме того, OpenPGP предоставляет функции кодирования и передачи ключей и сертификатов, хотя хранение и управление ключами выходят за рамки документа.

2.1. Конфиденциальность на основе шифрования

OpenPGP комбинирует шифрование с симметричным и (необязательно) открытым ключом для защиты конфиденциальности. При использовании открытых ключей объект сначала шифруется с использованием симметричного ключа. Каждый симметричный ключ применяется лишь 1 раз, для одного объекта. Для каждого объекта (иногда это называют сессией) генерируется новый «сеансовый ключ» в виде случайного значения. Поскольку сеансовый ключ применяется лишь однократно, он привязывается к сообщению и передаётся внутри него. Для защиты ключа применяется его шифрование с использованием открытого ключа получателя, как указано ниже.

1. Отправитель создаёт сообщение.
2. Передающая реализация OpenPGP генерирует для этого сообщения случайный сеансовый ключ.
3. Сеансовый ключ шифруется с открытым ключом получателя и помещается в начало сообщения.
4. Передающая реализация OpenPGP может сжать сообщение, а затем шифрует его с использованием ключа сообщения, выведенного из сеансового ключа. Зашифрованное сообщение формирует оставшуюся часть сообщения OpenPGP.
5. Принимающая реализация расшифровывает сеансовый ключ, применяя секретный ключ получателя.
6. Принимающая реализация OpenPGP расшифровывает сообщения, используя ключ сообщения, выведенный из сеансового ключа. Если сообщение было сжато, выполняется декомпрессия.

При симметричном шифровании используется похожий процесс, но сеансовый ключ шифруется с использованием симметричного алгоритма и ключа, выведенного из общего секрета.

Для сообщения может применяться защита конфиденциальности и цифровая подпись. Сначала для сообщения создаётся подпись, которая присоединяется к сообщению, затем сообщение вместе с подписью шифруется с симметричным ключом сообщения, выведенным из сеансового ключа. Затем сеансовый ключ шифруется с использованием открытого ключа и добавляется перед шифрованным блоком.

2.2. Аутентификация с помощью цифровых подписей

Для цифровых подписей применяется криптографическая хэш-функция и пригодный для подписей алгоритм с открытым ключом. Последовательность действий указана ниже.

1. Отправитель создаёт сообщение.
2. Передающая реализация OpenPGP генерирует хэш-значение (дайджест) для сообщения.
3. Передающая реализация генерирует подпись из дайджеста с использованием секретного ключа отправителя.
4. Подпись прикрепляется к сообщению или передаётся вместе с ним.
5. Принимающая реализация получает копию сообщения и его подпись.
6. Принимающая реализация генерирует новый хэш-дайджест для полученного сообщения и проверяет его, используя подпись сообщения. При совпадении значений сообщение считается подлинным.

2.3. Сжатие

Реализации OpenPGP **могут** поддерживать сжатие данных. Многие сообщения OpenPGP сжимаются. Разработчики реализаций с ограничениями, не желающие поддерживать сжатие, могут ограничиться поддержкой декомпрессии.

2.4. Преобразование в base64

Базовое представление OpenPGP для шифрованных сообщений, подписей, ключей и сертификатов - это поток произвольных октетов. Некоторые системы разрешают использование лишь блоков, состоящих из 7-битовых печатных символов. Для доставки произвольных двоичных октетов OpenPGP через каналы, не поддерживающие произвольные (raw) двоичные данные, определено кодирование пригодными для печати символами. Поток произвольных 8-битовых двоичных октетов преобразуется в печатные символы ASCII и помощью кодирования base64 в формате ASCII Armor (раздел 6).

Реализациям **следует** поддерживать преобразования base64.

2.5. Приложения, работающие только с подписями

OpenPGP предназначается для приложений, использующих шифрование и подписи, но имеется ряд случаев, когда нужна реализация лишь подписей. Хотя эта спецификация требует подписей и шифрования, разумно предположить, что существует подмножество реализаций, не соответствующих спецификации в части поддержки шифрования.

3. Форматы элементов данных

В этом разделе рассматриваются элементы данных, используемые OpenPGP.

3.1. Численные скаляры

Численные скаляры не имеют знака и всегда хранятся в формате big-endian. Если использовать обозначение n[k] для указания k-го октета, значением 2-октетного скаляра будет ((n[0] << 8) + n[1]), 4-октетного - ((n[0] << 24) + (n[1] << 16) + (n[2] << 8) + n[3]).

3.2. Целые числа с увеличенной разрядностью

Целые числа с увеличенной разрядностью (Multiprecision Integer или MPI) - это целые числа без знака, используемые для больших значений, таких как применяемые в криптографических расчётах. MPI состоит из двух частей - 2-октетный скаляр, указывающий число битов MPI, за которым следует строка октетов, содержащая значение числа (эти октеты образуют число в формате big-endian, которое можно преобразовать в MPI добавлением префикса, задающего размер). Например (в квадратных скобках указаны шестнадцатеричные значения), строка октетов [00 00] указывает MPI со значением 0, строка [00 01 01] - MPI со значением 1, [00 09 01 FF] - MPI со значением 511.

Для MPI применяются указанные ниже правила.

- Размер MPI в октетах составляет ((MPI.length + 7) / 8) + 2.
- Поле размера MPI указывает размер, начиная со старшего ненулевого бита. Таким образом MPI [00 02 01] не является корректным и следует использовать [00 01 01]. При синтаксическом анализе MPI в пакетах Key, Signature, Public Key Encrypted Session Key (PKESK) версии 6 реализация **должна** проверять соответствие указанного размера числу битов, начиная со старшего бита с ненулевым значением. При несовпадении пакет отвергается как некорректно сформированный.
- Неиспользуемые биты MPI **должны** иметь значение 0.

3.2.1. Применение MPI для кодирования других данных

Отметим, что в некоторых местах MPI применяются для кодирования нечисловых данных, таких как точки эллиптических кривых (EC, параграф 11.2) или строк октетов известного фиксированного размера (параграф 11.3). При передаче применяется такое же представление - 2 октета размера (в битах, начиная с первого ненулевого), затем наименьшая серия октетов, которая может представлять значение (нули в начале исключаются).

3.3. Идентификаторы и оттиски ключей

Идентификатор ключа (Key ID) - 8-октетный скаляр, указывающий ключ. Реализациям **не следует** считать Key ID уникальными. Уникальность оттиска (fingerprint) значительно вероятней уникальности Key ID. Оттиски и Key ID для ключа рассчитываются в зависимости от версии ключа. Формирование оттисков и Key ID описано в параграфе 5.5.4.

3.4. Текст

Если не указано иное, текст представляется в UTF-8 [RFC3629] с кодировкой [ISO10646].

3.5. Временные поля

Поля времени указываются 4-октетным целым числом без знака, содержащим число секунд, прошедших с 1 января 1970 г по времени UTC.

3.6. Связка ключей

Связка ключей (keyring) - это набор из 1 или нескольких ключей в файле или базе данных. Обычно это просто последовательный список ключей, но может быть и подходящая база данных. Детали связок ключей и других баз данных выходят за рамки этой спецификации.

3.7. Спецификатор преобразования строки в ключ (S2K)

Спецификаторы преобразования строки в ключ (string-to-key или S2K) служат для создания ключа шифрования/дешифрования из парольной фразы. Парольные фразы, требующие S2K, в настоящее время применяются лишь для шифрования секретной части приватных ключей и симметрично шифруемых сообщений.

3.7.1. Типы S2K

В настоящее время имеется 4 типа выделенных спецификаторов S2K и несколько резервных значений.

Таблица 1. Реестр типов OpenPGP S2K.

ID	*Тип S2K*	*Число октетов*	*Создание?*	*Документ*
0	Простое преобразование S2K	2	Нет	параграф 3.7.1.1
1	S2K с затравкой (salt)	10	Только при высокой энтропии строки	параграф 3.7.1.2
2	Резерв	-	Нет	
3	S2K с итерациями и затравкой	11	Да	параграф 3.7.1.3
4	Argon2	20	Да	параграф 3.7.1.4
100-110	Частное применение и эксперименты	-	Если нужно	

Типы спецификаторов S2K описаны в последующих параграфах. Если в столбце «Создание?» не указано значение «Да», запись S2K применяется лишь для чтения в целях совместимости с прежними версиями и её не следует использовать для генерации.

3.7.1.1. Простое преобразование S2K

Простое преобразование Simple S2K напрямую хэширует строку для создания данных ключа, как указано ниже.

Октет 0 0x00
Октет 1 Алгоритм хэширования

Simple S2K хэширует парольную фразу для создания сеансового ключа. Способ хэширования зависит от размера сеансового ключа (который зависит от используемого шифра) и размера вывода алгоритма хэширования. Если вывод хэширования превышает размер сеансового ключа используются старшие (слева) октеты хэш-значения. Если вывод хэширования меньше размера ключа, создаётся несколько экземпляров хэш-контекста, чтобы их было достаточно для создания требуемых данных ключа. Эти экземпляры предварительно заполняются 0, 1, 2, ... октетами нулей (первый экземпляр не загружается заранее, второй загружается с одним октетом нулей, третий - с двумя и т. д.). При хэшировании данные выдаются независимо в каждый контекст. Поскольку исходно контексты различаются, каждый будет давать свой результат хэширования. После хэширования парольной фразы выходные данные из всех контекстов объединяются (конкатенация, первый хэш слева) для создания данных ключа, а лишние октеты отбрасываются.

3.7.1.2. S2K с затравкой

Преобразование Salted S2K включает значение «затравки» (salt - некие произвольные данные), хэшируемое вместе с парольной фразой для предотвращения атак по словарю.

Октет 0 0x01
Октет 1 Алгоритм хэширования
Октеты 2-9 8-октетное значение затравки (salt)

Преобразование Salted S2K похоже на Simple S2K, отличаясь лишь тем, что на вход хэш-функции перед парольной фразой передаются 8 октетов затравки из S2K Specifier.

3.7.1.3. S2K с затравкой и итерациями

S2K с затравкой и итерациями включает затравку и счётчик октетов. Затравка объединяется с парольной фразой и полученное значение повторяется, затем хэшируется. Это увеличивает объем работы, которую злоумышленник должен выполнить для атаки по словарю.

Октет 0 0x01
Октет 1 Алгоритм хэширования
Октеты 2-9 8-октетное значение затравки (salt)
Октет 10 1-октетное кодированное значение счётчика

Счётчик кодируется в 1-октетное целое число, как показано ниже.

```
#define EXPBIAS 6
    count = ((Int32)16 + (c & 15)) << ((c >> 4) + EXPBIAS);
```

Это выражение описано в [C99], Int32 - тип для 32-битовых целых чисел, а переменная c - кодированное значение счётчика (октет 10).

В S2K с итерациями и затравкой парольная фраза и затравка хэшируются неоднократно. Общее число число хэшируемых октетов задаётся кодированным счётчиком в спецификаторе S2K. Отметим, что результирующее значение октета count указывает число хэшируемых октетов, а не число итераций.

Изначально организуется 1 или несколько контекстов хэширования (как и в других алгоритмах S2K) в зависимости от требуемого объёма данных для ключа. Затем затравка и следующая за ней парольная фраза многократно обрабатываются как входные данные для каждого контекста хэширования, пока не будет хэшировано заданное значением count число октетов. Вывод сокращается до требуемого числа октетов, если только это число не меньше исходного размера затравки и парольной фразы. Т. е. независимо от счётчика октетов на вход каждого контекста хэширования будет передана по меньшей мере 1 полная копия затравки и парольной фразы. По завершении хэширования из хэш-дайджестов формируются данные ключа как и в других алгоритмах S2K.

3.7.1.4. Argon2

В этом преобразовании S2K парольная фраза хэшируется с использованием Argon2, как указано в [RFC9106]. Это обеспечивает интенсивное использование памяти и дополнительную защиту парольных фраз от атак методом перебора (brute-force).

Октет 0 0x04
Октеты 1-16 16-октетное значение затравки
Октет 17 1-октетное значение числа проходов t
Октет 18 1-октетное значение степени распараллеливания p
Октет 19 1-октетное значение encoded_m, показатель степени для размера памяти

Значение затравки **следует** делать уникальным для каждой парольной фразы. Значения числа проходов t и степени распараллеливания p **должны** быть ненулевыми. Размер памяти m составляет 2^encoded_m кбайт (KiB) ОЗУ (RAM). Значение закодированного размера памяти **должно** составлять от 3+ceil(log_2(p)) до 31, так что декодированный размер m составит от 8*p до 2^31. Отметим, что размер памяти указывается в KiB, а не в октетах.

Argon2 вызывается с парольной фразой P, затравкой S, значениями t, p и m, как указано выше, требуемым размером ключа как длины метки T, значением 0x13 как номера версии v и Argon2id в качестве типа. Рекомендации для значений t, p и m приведены в разделе 4 [RFC9106]. Если рекомендованное значение m для данного применения не является степенью 2, **рекомендуется** округлить его вверх до степени 2 при условии сохранения приемлемой производительности, иначе округлить вниз с учётом того, что m должно быть не менее 8*p.

Например, при первом рекомендованном варианте (t=1, p=4, m=2^21) полный спецификатор S2K будет иметь вид

```
04 xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx 01 04 15
```

где XX - случайные значения затравки.

3.7.2. Применение S2K

Спецификаторы Simple S2K и Salted S2K уязвимы к атакам путём перебора (brute-force) при использовании строк с низкой энтропией, какbt обычно предоставляют пользователи. Кроме того, применение Simple S2K может приводить к повторному использованию ключа и вектора инициализации (IV) (см. параграф 5.3). Поэтому при генерации

спецификатора S2K реализации **недопустимо** применять Simple S2K. Более того, реализации не следует генерировать Salted S2K, пока нет уверенности в достаточной энтропии входной строки (например при генерации строки самой реализацией с использованием хорошего источника случайных значений).

Реализациям **рекомендуется** использовать Argon2. При недоступности Argon2 **можно** применять Iterated и Salted S2K, если обеспечивается большое число октетов и сильная парольная фраза. Однако эти методы не обеспечивают достаточную стойкость памяти, в отличие от Argon2.

3.7.2.1. Шифрование секретного ключа

Первый октет, следующий за материалом открытого ключа в пакете Secret Key (параграф 5.5.3), указывает защищён ли материал секретного ключа парольной фразой и как. Этот октет называют октетом применения S2K (S2K usage). Если этот октет имеет значение 0, данные секретного ключа не защищены, иное значение указывает, как использовать парольную фразу для разблокирования секретного ключа.

Реализации, предшествующие [RFC2440], указывали защищённый ключ, сохраняя идентификатор симметричного алгоритма шифрования (Symmetric Cipher Algorithm ID, параграф 9.3) в октете применения S2K. В этом случае для преобразования парольной фразы в ключ указанного алгоритма шифрования всегда применялась хэш-функция MD5. Более поздние реализации указывают защищённый секретный ключ, сохраняя одно из специальных значений 253 (AEAD), 254 (CFB) или 255 (MalleableCFB) в октете применения S2K. За этим октетом сразу же следует набор полей, описывающих преобразование парольной фразы в симметричный ключ, с помощью которого можно разблокировать секретный материал, а также другие параметры, относящиеся к используемому типу шифрования.

Формат передачи зависит от версии вложенного пакета OpenPGP. В таблице ниже приведена сводка особенностей, описанных в параграфе 5.5.3. В этой таблице check(x) обозначает «2-октетную контрольную сумму», которая представляет собой сумму всех октетов значения x mod 65536. Параметры info и packetprefix подробно описаны в параграфе 5.5.3. Заголовок столбца «Создаётся?» сокращён до «Созд?».

Таблица 2. Реестр шифрования секретных ключей OpenPGP (октет использования S2K).

Октет применения S2K	***Сокращение***	***Поля параметров шифрования***	***Шифрование***	***Созд?***	
0	Unprotected	-	Ключи v3, v4: [cleartext secrets check(secrets)], ключи v6: [cleartext secrets]	Да	
Идентификатор известного симметричного шифра (см. параграф 9.3)	LegacyCFB	IV	CFB(MD5(passphrase), secrets	Нет	check(secrets))
253	AEAD	Params-length (только v6), cipher-algo, AEAD-mode, S2K-specifier-length (только v6), S2K-specifier, nonce	AEAD(HKDF(S2K(pass phrase), info), secrets, packetprefix)	Да	
254	CFB	Params-length (только v6), cipher-algo, S2K-specifier-length (только v6), S2K-specifier, IV	CFB(S2K(passphrase), secrets	Да	SHA1(secrets))
255	MalleableCFB	Cipher-algo, S2K-specifier, IV	CFB(S2K(passphrase), secrets	Нет	check(secrets))

При выдаче секретного ключа (с защитой или без таковой) реализация **должна** выдавать данные лишь для строк со значением «Да» в столбце «Созд?». Прочие строки указаны лишь для совместимости с прежними версиями (чтение ключей версии 4 и более ранних) и их использование для генерации новых ключей **недопустимо**. Секретные ключи версии 6 с таким форматом **должны** отвергаться. Отметим, что по сравнению с секретными ключами версии 4 параметры защищённых паролем секретных ключей версии 6 сохраняются с дополнительной парой размеров, каждый из которых занимает 1 октет.

Argon2 применяется лишь при аутентифицированном шифровании со связанными данными (Authenticated Encryption with Associated Data или AEAD) (октет применения S2K 253). Реализациям **недопустимо** создавать и они **должны** отвергать как некорректно сформированные любые секретные ключи, где октет применения S2K не указывает AEAD (253), а тип спецификатора S2K - Argon2.

3.7.2.2. Шифрование сообщений с симметричным ключом

OpenPGP может создавать пакет с сеансовым ключом, зашифрованным с симметричным ключом (Symmetric Key Encrypted Session Key или SKESK) в начале сообщения. Это позволяет применять спецификаторы S2K для преобразования парольной фразы или создания сообщений, содержащих смесь пакетов SKESK и PKESK. Такое сообщение можно расшифровать с помощью парольной фразы или ключа из асимметричной пары.

Реализации, предшествующие [RFC2440], всегда использовали международный алгоритм шифрования данных (International Data Encryption Algorithm или IDEA) с преобразованием Simple S2K при шифровании сообщений с симметричным алгоритмом (параграф 5.7). Генерировать IDEA **недопустимо**, но **можно** воспринимать для совместимости со старыми версиями.

4. Синтаксис пакетов

В этом разделе описаны пакеты, применяемые OpenPGP.

4.1. Обзор

Сообщение OpenPGP собирается из нескольких записей, которые обычно называют пакетами. Пакет - это блок данных (chunk) с Type ID, определяющим назначение. Сообщение OpenPGP, связка ключей, сертификат, отдельная подпись и т. п. состоят из множества пакетов. Некоторые из этих пакетов могут содержать другие пакеты OpenPGP (например,

сжатый пакет данных, который после декомпрессии будет содержать пакеты OpenPGP). Каждый пакет состоит из заголовка, за которым следует тело пакета. Размер пакетов является переменной величиной.

При обработке потока пакетов сведения о размере в заголовке каждого пакета являются каноническим источником для определения границ пакета. Реализация, обрабатывающая поток пакетов и желающая найти следующий пакет, **должна** искать его по точному смещению, указанному в заголовке предыдущего пакета.

Некоторые пакеты включают внутренние индикаторы размера (например, субполя в пакете). Когда индикатор размера в субполе внутри пакета подразумевает включение октетов за пределами диапазона, указанного в заголовке пакета, анализатор **должен** прервать обработку без записи за пределами указанного диапазона и считать пакет сформированным некорректно и непригодным. Реализациям **недопустимо** считать октеты за пределами диапазона, указанного в заголовке пакета, частью содержимого пакета.

4.2. Заголовок пакета

Первый октет пакета указывает формат остальной части заголовка и кодирует идентификатор типа пакета (Packet Type ID), как описано в разделе 5. Оставшаяся часть заголовка указывает размер пакета. Имеется два формата пакетов: 1) пакет (текущего) OpenPGP, заданный этим документом и его предшественниками [RFC4880] и [RFC2440], 2) унаследованный (Legacy) пакет, используемых в спецификациях, предшествующих IETF OpenPGP.

Старший бит октета называется битом 7 и маской для него служит шестнадцатеричное значение 0x80. Кодированный идентификатор пакета имеет вид

```
+---------------+
|7 6 5 4 3 2 1 0|
+---------------+
```

Формат OpenPGP

Бит 7 - 1
Бит 6 - 1
Биты 5 - 0 - Packet Type ID

Унаследованный формат

Бит 7 - 1
Бит 6 - 0
Биты 5 - 2 - Packet Type ID
Биты 2 - 0 - length-type

Бит 6 в первом октете заголовка пакета указывает формат OpenPGP или Legacy. Пакеты Legacy **можно** воспринимать для обеспечения функциональной совместимости и доступа к архивированным данным. Пакеты Legacy **не следует** использовать для генерации новых данных, если нет уверенности в том, что получатель поддерживает лишь этот формат. Такая ситуация крайне маловероятна, поскольку формат Legacy был признан устаревшим в 1998 г. [RFC2440].

Реализация, принимающая и распространяющая существовавшие ранее данные OpenPGP, например, переносимые открытые ключи (Transferable Public Key), может столкнуться с пакетами в формате Legacy. Такие пакеты **можно** распространять далее в их формате Legacy или преобразовать в текущий формат OpenPGP перед распространением.

Отметим, что в заголовках пакетов Legacy идентификатор типа пакета имеет лишь 4 бита, поэтому может указывать только типы меньше 16, а заголовки формата OpenPGP могут кодировать идентификаторы вплоть до 63.

4.2.1. Размеры пакетов OpenPGP

Для пакетов OpenPGP имеется 4 возможных способа декодирования размера.

1. 1-октетный заголовок Body Length кодирует размеры пакетов до 191 октетов.
2. 2-октетный заголовок Body Length кодирует размеры пакетов от 192 до 8383 октетов.
3. 5-октетный заголовок Body Length кодирует размеры пакетов до 4294967295 (0xFFFFFFFF) октетов (фактически для кодирования применяется 4-октетное скалярное значение).
4. Если размер пакета не известен эмитенту заранее, применяются заголовки Partial Body Length, кодирующие пакет неопределённого размера, фактически превращая его в поток.

4.2.1.1. 1-октетный заголовок

1-октетный заголовок Body Length кодирует размер от 0 до 191 октетов. Этот тип заголовка распознаётся по 1-октетному значению меньше 192. Размер тела пакета определяется как

```
bodyLen = 1st_octet;
```

4.2.1.2. 2-октетный заголовок

2-октетный заголовок Body Length кодирует размер от 192 до 8383 и распознаётся по первому октету со значением от 192 до 223. Размер тела пакета определяется как

```
bodyLen = ((1st_octet - 192) << 8) + (2nd_octet) + 192
```

4.2.1.3. 5-октетный заголовок

5-октетный заголовок Body Length состоит из октета со значением 255, за которым следует 4-октетный скаляр. Размер тела пакета определяется как

```
bodyLen = (2nd_octet << 24) | (3rd_octet << 16) |
          (4th_octet << 8)  | 5th_octet
```

Этот базовый набор 1-, 2- и 5-октетных заголовков размера применятся также внутри некоторых пакетов.

4.2.1.4. Заголовок Partial Body Length

Заголовок Partial Body Length - это 1 октет, кодирующий размер лишь части пакета данных, который является степенью 2 от 1 до 1073741824 (2^30). Заголовок распознаётся по значению октета не меньше 224, но меньше 255. Partial Body Length имеет значение

```
partialBodyLen = 1 << (1st_octet & 0x1F);
```

За каждым заголовком Partial Body Length следует часть данных тела пакета, размер которой указывает этот заголовок. За этой частью данных следует другой заголовок размера (1, 2, 5 октетов или Partial Body Length). В качестве последнего заголовка размера **недопустимо** использовать Partial Body Length (эти заголовки могут применяться только для нефинальных частей пакета). Отметим, что последний заголовок Body Length может указывать размер 0.

Реализация **может** использовать Partial Body Length для пакетов данных, независимо от того, являются ли они литеральными, сжатыми или шифрованными. Первый частичный размер **должен** быть не менее 512 октетов. **Недопустимо** использовать Partial Body Length для пакетов иного типа.

4.2.2. Размер пакетов унаследованных форматов

Бит 6 со значением 0 в первом октете пакета указывает унаследованный формат (Legacy). Биты 1 и 0 в первом октете пакета Legacy являются полем length-type (размер-тип), значение которого описано ниже.

0 Пакет имеет 1-октетный размер, размер заголовка - 2 октета.
1 Пакет имеет 2-октетный размер, размер заголовка - 3 октета.
2 Пакет имеет 4-октетный размер, размер заголовка - 5 октетов.
3 Заголовок имеет размер 1 октет, размер пакета не определён и реализация должна определить его. Если пакет является файлом, это означает, что пакет простирается до конца файла. Заголовки формата OpenPGP имеют механизм для точного кодирования данных неопределённого размера. Реализации **недопустимо** создавать пакеты формата Legacy с неопределённым размером. Реализация **может** интерпретировать пакет неопределённого размера с форматом Legacy для работы со старыми данными или данными, созданными унаследованной системой (до [RFC2440]).

4.2.3. Примеры размера пакетов

Ниже приведены примеры кодирования размера в пакетах формата OpenPGP.

- Пакет с телом в 100 октетов может кодировать размер в одном октете 0x64, за которым следуют данные.
- Пакет с телом в 1723 октета может кодировать размер в 2 октетах 0xC5 и 0xFB, за которыми следуют данные.
- Пакет с телом в 100000 октетов может кодировать размер в 5 октетах 0xFF, 0x00, 0x01, 0x86, 0xA0.

Можно также закодировать потоки октетов:

- 0xEF - первые 32768 октетов данных;
- 0xE1 - следующие 2 октета данных;
- 0xE0 - следующий 1 октет данных;
- 0xF0 - следующие 65536 октетов данных;
- 0xC5, 0xDD - последние 1693 октета данных.

Это лишь один из возможных вариантов кодирования и возможно много вариаций размера с использованием заголовков Partial Body Length, тогда как заголовок Body Length кодирует размер последней части данных. Отметим, что во всех случаях размер пакета равен сумме размеров заголовка и тела пакета.

4.3. Критичность пакета

Пространство идентификаторов типа пакета (Packet Type ID) разделено на 2 части - критичные и некритичные пакеты. Если реализация встречает в последовательности критичный пакет неизвестного типа, она **должна** отклонить всю последовательность пакетов (см. раздел 10). Неизвестные некритичные пакеты **должны** игнорироваться. Критичны пакеты с идентификаторами типа от 0 до 39, а пакеты с идентификаторами 40 - 63 некритичны.

5. Типы пакетов

Определённые в настоящее время типы пакетов указаны в таблице ниже.

Таблица 3. Реестр типов пакетов OpenPGP.

ID	*Критичный*	*Описание*	*Сокращение*	*Документ*
0	Да	Резерв (применять этот тип недопустимо)		
1	Да	Сеансовый ключ, зашифрованный с открытым ключом	PKESK	параграф 5.1
2	Да	Подпись	SIG	параграф 5.2
3	Да	Сеансовый ключ, зашифрованный с симметричным ключом	SKESK	параграф 5.3
4	Да	One-Pass Signature Packet	OPS	параграф 5.4
5	Да	Секретный ключ	SECKEY	параграф 5.5.1.3
6	Да	Открытый ключ	PUBKEY	параграф 5.5.1.1
7	Да	Секретный субключ	SECSUBKEY	параграф 5.5.1.4
8	Да	Сжатые данные	COMP	параграф 5.6
9	Да	Данные, зашифрованные с симметричным ключом	SED	параграф 5.7
10	Да	Маркер	MARKER	параграф 5.8
11	Да	Литеральные данные	LIT	параграф 5.9
12	Да	Доверие	TRUST	параграф 5.10
13	Да	Идентификатор пользователя	UID	параграф 5.11
14	Да	Открытый субключ	PUBSUBKEY	параграф 5.5.1.2
17	Да	Атрибуты пользователя	UAT	параграф 5.12
18	Да	Данные с симметричным шифрованием и защитой целостности	SEIPD	параграф 5.13
19	Да	Резерв (ранее пакет с кодом обнаружения изменений)		параграф 5.13.1
20	Да	Резерв		

21	Да	Заполнение	PADDING	параграф 5.14
22-39	Да	Невыделенные критичные пакеты		
40-59	Нет	Невыделенные некритичные пакеты		
60-63	Нет	Приватное использование и эксперименты		

Сокращённые обозначения из соответствующей колонки применяются в этом документе для компактности и могут также использоваться в реализациях с возможностями отладки или проверки потоков пакетов OpenPGP.

5.1. Сеансовый ключ, зашифрованный с открытым ключом (Type ID 1)

Перед контейнером шифрования (шифрованное сообщение), т. е. пакетом с симметричным шифрованием и защитой целостности (Symmetrically Encrypted and Integrity Protected Data или SEIPD) или пакетом старых данных с симметричным шифрованием (Symmetrically Encrypted Data или SED), могут присутствовать пакеты PKESK и/или SKESK. Сообщение шифруется с сеансовым ключом, а сам этот ключ шифруется и сохраняется в пакете(ах) Encrypted Session Key. Перед контейнером шифрования размещается 1 пакет зашифрованного с открытым ключом сеансового ключа (Public Key Encrypted Session Key) для каждого ключа OpenPGP Key, использованного для шифрования сообщения. Получатель сообщения находит сеансовый ключ, зашифрованный с открытым ключом получателя, расшифровывает сеансовый ключ и использует его для расшифровки сообщения.

Тело этого пакета начинается с 1-октетного числа, указывающего номер версии для типа пакета. В настоящее время определены версии 3 и 6. Остальная часть пакета зависит от версии.

Версии различаются способом идентификации ключа получателя и кодированного содержимого. Версия пакета PKESK должна соответствовать версии пакета SEIPD (параграф 10.3.2.1). Новые версии пакетов PKESK следует регистрировать в реестре, описанном в параграфе 10.3.2.1.

5.1.1. Формат пакета версии 3

Пакет PKESK версии 3 предшествует пакету SEIPD v1 (см. параграф 5.13.1). В старых данных этот пакет иногда может появляться перед устаревшим пакетом SED (см. параграф 5.7). Пакету PKESK v3 **недопустимо** быть перед пакетом SEIPD v2 (см. параграф 10.3.2.1).

Состав пакета PKESK v3 указан ниже.

- 1 октет номера версии со значением 3.
- 8-октетное значение Key ID открытого ключа, использованного для шифрования сеансового ключа. Если сеансовый ключ зашифрован с субключом, применяется Key ID этого субключа вместо Key ID основного ключа. Идентификатор ключа может состоять из нулей для анонимного получателя (см. параграф 5.1.8).
- 1 октет, указывающий использованный алгоритм с открытым ключом.
- Последовательность значений зашифрованного сеансового ключа, зависящая от алгоритма (см. ниже).

Алгоритму шифрования с открытым ключом (см. ниже) передаются два значения:

- сеансовый ключ;
- 1-октетный идентификатор алгоритма, указывающий алгоритм симметричного шифрования, использованный для зашифровки пакета SEIPD v1, описанного ниже.

5.1.2. Формат пакета версии 6

Пакет PKESK версии 6 предшествует пакету SEIPD v2 (параграф 5.13.2). Пакету PKESK v6 **недопустимо** быть перед пакетом SEIPD v1 или устаревшим пакетом SED (параграф 10.3.2.1).

Состав пакета PKESK v6 указан ниже.

- 1 октет номера версии со значением 6.
- 1 октет размера двух следующих полей. Это поле может иметь значение 0, если поля номера версии ключа и оттиска опущены для анонимного получателя (параграф 5.1.8).
- 1 октет номера версии ключа.
- Оттиск (fingerprint) открытого ключа или субключа, с которым шифруется сеансовый ключ. Отметим, что размер оттиска ключа N для версии 4 составляет 20 октетов, для версии 6 - 32.
- 1 октет, указывающий использованный алгоритм с открытым ключом.
- Последовательность значений зашифрованного сеансового ключа, зависящая от алгоритма (см. ниже).

Сеансовый ключ шифруется в соответствии с применяемым алгоритмом с открытым ключом, как описано ниже. Идентификатор симметричного алгоритма шифрования не передаётся алгоритму с открытым ключом для пакета PKESK v6, поскольку он включён в пакет SEIPD v2.

5.1.3. Поля для шифрования RSA

- MPI зашифрованного с помощью RSA значения m^e mod n.

Для получения значения m в приведённой выше формуле сначала объединяются (конкатенация) значения:

- 1-октетного идентификатора алгоритма, если он был передан (в случае пакета PKESK v3);
- сеансового ключа;
- 2-октетной контрольной суммы сеансового ключа (сумма октетов сеансового ключа по модулю 65536).

Затем указанные выше значения кодируются с использованием блочного кодирования PKCS#1 EME-PKCS1-v1_5, как описано в п. 2 параграфа 7.2.1 в [RFC8017] (см. также параграф 12.1.1). При декодировании m в процессе дешифровки реализации следует выполнять п. 3 из параграфа 7.2.2 в [RFC8017] (см. также параграф 12.1.2).

Когда реализация формирует несколько пакетов с одним сеансовым ключом, создавая сообщение, которое может расшифровываться с несколькими ключами, она **должна** применять кодирование PKCS#1 для каждого ключа. Это защищает от атак, подобных описанным в [HASTAD].

5.1.4. Поля для шифрования Elgamal

- MPI зашифрованного с помощью Elgamal (Diffie-Hellman) значения g^k mod p.
- MPI зашифрованного с помощью Elgamal (Diffie-Hellman) значения m * y^k mod p.

Для получения значения m в приведённой выше формуле сначала объединяются (конкатенация) значения:

- 1-октетного идентификатора алгоритма, если он был передан (в случае пакета PKESK v3);
- сеансового ключа;
- 2-октетной контрольной суммы сеансового ключа (сумма октетов сеансового ключа по модулю 65536).

Затем указанные выше значения кодируются с использованием блочного кодирования PKCS#1 EME-PKCS1-v1_5, как описано в п. 2 параграфа 7.2.1 в [RFC8017] (см. также параграф 12.1.1). При декодировании m в процессе дешифровки реализации следует выполнять п. 3 из параграфа 7.2.2 в [RFC8017] (см. также параграф 12.1.2).

Когда реализация формирует несколько пакетов с одним сеансовым ключом, создавая сообщение, которое может расшифровываться с несколькими ключами, она **должна** применять кодирование PKCS#1 для каждого ключа. Это защищает от атак, подобных описанным в [HASTAD].

Реализации **недопустимо** генерировать пакеты ElGamal PKESK v6.

5.1.5. Поля для шифрования ECDH

- MPI точки эллиптической кривой (EC), представляющей эфемерный открытый ключ в связанном с кривой формате, как указано в параграфе 9.2.
- 1 октет размера, за которым следует симметричный ключ, закодированный, как указано в параграфе 11.5.

5.1.6. Поля для шифрования X25519

- 32 октета, представляющие эфемерный открытый ключ X25519.
- 1 октет размера последующих полей.
- 1-октет идентификатора алгоритма, если он был передан (в случае пакета PKESK v3).
- Зашифрованный сеансовый ключ.

Детали расчёта эфемерного открытого ключа и общего секрета приведены в параграфе 6.1 [RFC7748]. Затем применяется основанная на HMAC функция вывода ключа (Key Derivation Function или HKDF) [RFC5869] с SHA256 [RFC6234], информационным параметром «OpenPGP X25519» и без затравки. На вход HKDF передаётся конкатенация трёх значений:

- 32 октета эфемерного открытого ключа X25519 для этого пакета;
- 32 октета открытого ключевого материала получателя;
- 32 октета общего секрета.

Созданный с помощью HKDF ключ служит для шифрования сеансового ключа с AES-128, как задано в [RFC3394].

Отметим, что в отличие от Elliptic Curve Diffie-Hellman (ECDH) к сеансовому ключу не добавляется контрольной суммы или заполнения перед упаковкой (wrapping) ключа. В отличие от других алгоритмов с открытым ключом в случае пакета PKESK v3 идентификатор симметричного алгоритма не шифруется. Вместо этого он в открытом виде помещается перед зашифрованным сеансовым ключом. В этом случае **должен** применяться симметричный алгоритм AES-128, AES-192 или AES-256 (идентификаторы 7, 8, 9, соответственно).

5.1.7. Поля для шифрования X448

- 56 октетов, представляющих эфемерный открытый ключ X448.
- 1 октет размера последующих полей.
- 1-октет идентификатора алгоритма, если он был передан (в случае пакета PKESK v3).
- Зашифрованный сеансовый ключ.

Детали расчёта эфемерного открытого ключа и общего секрета приведены в параграфе 6.2 [RFC7748]. Затем применяется HKDF [RFC5869] с SHA512 [RFC6234], информационным параметром «OpenPGP X448» и без затравки. На вход HKDF передаётся конкатенация трёх значений:

- 56 октетов эфемерного открытого ключа X448 для этого пакета;
- 56 октетов открытого ключевого материала получателя;
- 56 октетов общего секрета.

Созданный с помощью HKDF ключ служит для шифрования сеансового ключа с AES-256, как задано в [RFC3394].

Отметим, что в отличие от ECDH, к сеансовому ключу не добавляется контрольной суммы или заполнения перед упаковкой (wrapping) ключа. В отличие от других алгоритмов с открытым ключом в случае пакета PKESK v3 идентификатор симметричного алгоритма не шифруется. Вместо этого он в открытом виде помещается перед зашифрованным сеансовым ключом. В этом случае **должен** применяться симметричный алгоритм AES-128, AES-192 или AES-256 (идентификаторы 7, 8, 9, соответственно).

5.1.8. Замечания по PKESK

Реализация **может** воспринимать или использовать Key ID, состоящий только из нулей, или отсутствие оттиска для сокрытия предусмотренного ключа расшифровки. В этом случае принимающая реализация будет пробовать все доступные секретные ключи, проверяя пригодность расшифрованного сеансового ключа. Этот формат помогает сократить возможности анализа трафика сообщений.

5.2. Подпись (Type ID 2)

Пакет подписи (Signature) описывает связь между неким открытым ключом и некими данными. Наиболее распространены подписи файлов и блоков текста, а также подписи, удостоверяющие идентификатор пользователя (User ID).

Определены 3 версии пакетов Signature. Версия 3 предоставляет основные сведения о подписи, а версии 4 и 6 обеспечивают расширенный формат с субпакетами, в которых можно указать дополнительную информацию о подписи. В силу исторических причин пакеты версий 1, 2, 5 не определены. Любую новую версию пакетов Signature следует регистрировать в реестре, заданном в параграфе 10.3.2.2.

Реализация **должна** создавать подписи версии 6 при подписании с ключом версии 6 и подписи версии 4 при подписании с ключом версии 4. Реализациям **недопустимо** создавать подписи версии 3, но **можно** воспринимать такие подписи. В параграфе 10.3.2.2 более подробно рассматривается соответствие версий ключей и подписей.

5.2.1. Типы подписей

Существует множество вариантов подписей, указываемых идентификатором типа (Signature Type ID) в каждой подписи. Отметим, что нечёткость различий не является недостатком системы и относится, скорее, к её особенностям. Поскольку OpenPGP возлагает окончательную проверку подписи на получателя, может оказаться, что небрежное действие одного подписавшего будет более строгим, чем осознанное действие другого. Подробные сведения о расчёте и проверке подписей каждого типа приведено в параграфе 5.2.4.

Таблица 4. Реестр типов подписей OpenPGP.

ID	*Имя*	*Документ*
0x00	Binary Signature (подпись двоичного документа)	параграф 5.2.1.1
0x01	Text Signature (подпись текстового документа)	параграф 5.2.1.2
0x02	Standalone Signature (автономная подпись)	параграф 5.2.1.3
0x10	Generic Certification Signature (базовая сертификационная подпись)	параграф 5.2.1.4
0x11	Persona Certification Signature (сертификационная подпись лица)	параграф 5.2.1.5
0x12	Casual Certification Signature (небрежная сертификационная подпись)	параграф 5.2.1.6
0x13	Positive Certification Signature (подпись положительной сертификации)	параграф 5.2.1.7
0x18	Subkey Binding Signature (подпись привязки субключа)	параграф 5.2.1.8
0x19	Primary Key Binding Signature (подпись привязки первичного ключа)	параграф 5.2.1.9
0x1F	Direct Key Signature (подпись прямого ключа)	параграф 5.2.1.10
0x20	Key Revocation Signature (подпись отзыва ключа)	параграф 5.2.1.11
0x28	Subkey Revocation Signature (подпись отзыва субключа)	параграф 5.2.1.12
0x30	Certification Revocation Signature (подпись отзыва сертификата)	параграф 5.2.1.13
0x40	Timestamp Signature (подпись временной метки)	параграф 5.2.1.14
0x50	Third-Party Confirmation Signature (подпись стороннего подтверждения)	параграф 5.2.1.15
0xFF	Резерв	параграф 5.2.1.16

Более подробные описания всех типов подписей даны в последующих параграфах.

5.2.1.1. Подпись двоичного документа (Type ID 0x00)

Указывает, что подписавшая сторона создала документ и владеет им или удостоверяет, что документ не был изменён.

5.2.1.2. Подпись текста канонического документа (Type ID 0x01)

Указывает, что подписавшая сторона создала документ и владеет им или удостоверяет, что документ не был изменён. Подпись рассчитывается для текстовых данных с приведением перевода строк к форме <CR><LF>.

5.2.1.3. Автономная подпись (Type ID 0x02)

Эта подпись представляет лишь подпись содержимого своего субпакета и рассчитывается аналогично подписи двоичного документа нулевого размера. Автономные подписи версии 3 **недопустимо** создавать и они **должны** игнорироваться.

5.2.1.4. Базовая сертификационная подпись (Type ID 0x10) пакета с User ID и открытым ключом

Эмитент сертификации не делает каких-либо конкретных заявлений о степени проверки того, что владелец ключа действительно является лицом, указанным идентификатором пользователя (User ID).

5.2.1.5. Сертификационная подпись лица (Type ID 0x11) для пакета с User ID и открытым ключом

Эмитент сертификации не проверял заявление о принадлежности ключа пользователю с указанным User ID.

5.2.1.6. Небрежная сертификационная подпись (Type ID 0x12) пакета с User ID и открытым ключом

Эмитент сертификации выполнил некоторую второстепенную (casual) проверку заявления об идентичности.

5.2.1.7. Позитивная сертификационная подпись (Type ID 0x13) пакета с User ID и открытым ключом

Эмитент сертификации выполнил существенную проверку заявления о тождественности. Большинство реализаций OpenPGP выпускает подписи ключей в форме базовых сертификатов (Type ID 0x10), некоторые реализации могут применять сертификаты 0x11-0x13, но мало кто различает эти типы.

5.2.1.8. Подпись привязки субключа (Type ID 0x18)

Это заявление ключа подписи верхнего уровня, указывающее владение субключом. Подпись рассчитывается непосредственно для первичного ключа и субключа, но не для User ID или иных пакетов. Подпись, привязывающая субключ подписи, **должна** иметь субпакет Embedded Signature, содержащий подпись 0x19, созданную с субключом подписи для основного ключа и субключа.

5.2.1.9. Подпись привязки первичного ключа (Type ID 0x19)

Это заявление ключа подписи, указывающее, что им владеет первичный ключ. Подпись создаётся так же, как Subkey Binding (Type ID 0x18), непосредственно для первичного ключа и субключа, но не для User ID или иных пакетов.

5.2.1.10. Прямая подпись ключа (Type ID 0x1F)

Эта подпись создаётся непосредственно для ключа, привязывает сведения из субпакетов Signature к ключу и подходит для использования в субпакетах, предоставляющих информацию о ключе, таких как Key Flags или Revocation Key (устарел). Подпись подходит также для заявлений, которые органы, не сертифицирующие себя, хотят сделать для ключа, а не привязки ключа к имени.

5.2.1.11. Подпись отзыва ключа (Type ID 0x20)

Эта подпись создаётся непосредственно для отзываемого ключа, который не будет применяться. Действительными следует считать лишь подписи отзыва, сделанные с отзываемым ключом или Revocation Key (устарел).

5.2.1.12. Подпись отзыва субключа (Type ID 0x28)

Эта подпись создаётся непосредственно для первичного ключа и отзываемого субключа, который больше не будет применяться. Действительными подписями отзыва следует считать лишь подписи, сделанные с первичным ключом, связанным с отзываемым субключом, или Revocation Key (устарел).

5.2.1.13. Подпись отзыва сертификата (Type ID 0x30)

Эта подпись отзывает прежнюю подпись сертификата User ID (Type ID 0x10 - 0x13) или подпись Direct Key (Type ID 0x1F). Её следует создавать с тем же ключом, который применялся для отзываемой подписи, или Revocation Key (устарел). Подпись рассчитывается для тех же данных, что и отзываемый сертификат и ей следует иметь более позднюю дату, нежели в отзываемом сертификате.

5.2.1.14. Подпись временной метки (Type ID 0x40)

Эта подпись значима только для содержащейся в ней временной метки.

5.2.1.15. Подпись стороннего подтверждения (Type ID 0x50)

Это подпись другого пакета OpenPGP Signature (по аналогии с нотариальным заверением подписания данных). В подпись Third-Party Confirmation **следует** включать субпакет Signature Target, указывающий подтверждаемую подпись.

5.2.1.16. Резерв (Type ID 0xFF)

Реализации **недопустимо** создавать подписи этого типа или подтверждать такие подписи (см. параграф 5.2.4.1).

5.2.2. Формат пакета подписи версии 3

Тело пакета подписи версии 3 содержит:

- 1 октет номера версии (3);
- 1 октет размера указанного ниже хэшированного материала (должен иметь значение 5):
  - 1 октет Signature Type ID;
  - 4 октета времени создания;
- 8 октетов Key ID подписавшего;
- 1 октет алгоритма с открытым ключом;
- 1 октет алгоритма хэширования;
- 2 октета, содержащие 16 левых битов подписанного хэш-значения;
- 1 или несколько MPI, образующих подпись (зависит от алгоритма, как описано ниже).

Конкатенация подписываемых данных, типа подписи и времени создания из пакета Signature (5 дополнительных октетов) хэшируется и результат используется в алгоритме подписи. 16 старших битов (два первых октета) хэш-значения включаются в пакет Signature, чтобы обеспечить возможность отклонения некоторых непригодных подписей при проверке.

Для подписей RSA от алгоритма зависит значение MPI в подписи (m^d mod n), для DSA - MPI значений DSA r и s.

Расчёт подписи основан не хэше подписываемых данных, как описано выше. Детали расчёта различаются для подписей DSA и RSA, как описано в параграфах 5.2.3.1 и 5.2.3.2.

5.2.3. Формат пакетов подписи версий 4 и 6

Тело пакетов Signature версий 4 и 6 содержит указанные ниже поля.

- 1 октет номера версии (4 или 6 в соответствии с версией).
- 1 октет Signature Type ID.
- 1 октет алгоритма с открытым ключом.
- 1 октет алгоритма хэширования.
- Скалярный счётчик октетов данных хэшированных субпакетов, размещённых за этим полем. Для версии 4 это поле имеет размер 2 октета, для версии 6 - 4. Поле указывает число октетов во всех хэшированных пакетах и указатель, увеличенный на это число, позволяет пропустить хэшированные субпакеты.
- Необязательный набор данных хэшированных субпакетов.
- Скалярный счётчик октетов данных нехэшированных субпакетов, размещённых за этим полем. Для версии 4 это поле имеет размер 2 октета, для версии 6 - 4. Поле указывает число октетов во всех нехэшированных пакетах и указатель, увеличенный на это число, позволяет пропустить нехэшированные субпакеты.
- Необязательный набор данных нехэшированных субпакетов.
- 2 октета, содержащие 16 левых битов подписанного хэш-значения.
- В подписях версии 6 поле переменного размера, содержащее:
  - 1 октет размера затравки (size), значение которого **должно** совпадать с заданным для алгоритма хэширования (таблица 23);
  - затравку - случайное значение указанного размера.
- 1 или несколько MPI, образующих подпись, или октеты без MPI (эта часть зависит от алгоритма)[1].

5.2.3.1. Зависящее от алгоритма поле подписей RSA

- Значение MPI подписи RSA (m^d mod n).

В подписях RSA хэш-значение кодируется с использованием PKCS#1 типа EMSA-PKCS1-v1_5, как описано в параграфе 9.2 [RFC8017] (см. также параграф 12.1.3). Это требует вставки хэш-значения в структуру ASN.1 как строки октетов. В структуру также включается идентификатор объекта (object identifier или OID) для алгоритма хэширования (таблица 24).

5.2.3.2. Зависящие от алгоритма поля подписей DSA и ECDSA

- MPI для r DSA или ECDSA.
- MPI для s DSA или ECDSA.

Подписи версии 3 **недопустимо** создавать и использовать с алгоритмами подписи на основе эллиптических кривых (Elliptic Curve Digital Signature Algorithm или ECDSA).

В подписях DSA **должен** применяться алгоритм хэширования с размером дайджеста не меньше числа битов q, группы, создаваемой значением генератора ключа DSA. Если выходной размер выбранного хэша больше числа битов q, результат хэширования отсекается справа до размера q. Этот (возможно, усечённый) результат функции хэширования рассматривается как число и применяется напрямую в алгоритме подписи DSA.

В подписях ECDSA **должен** применяться алгоритм хэширования с размером дайджеста не меньше fsize для кривой (параграф 9.2), кроме случая NIST P-521, где **должен** применяться алгоритм с размером хэша не менее 512 битов.

5.2.3.3. Зависящие от алгоритма поля подписей EdDSALegacy (устарели)

- Два значения MPI, содержимое и формат которых зависят от выбора кривой (см. параграф 9.2.1).

Подписи версии 3 **недопустимо** создавать и использовать с EdDSALegacy.

В подписях EdDSALegacy **должен** применяться алгоритм хэширования с размером дайджеста не меньше значения fsize для кривой (параграф 9.2). Проверяющая реализация **должна** отвергать подписи EdDSALegacy, использующие алгоритм хэширования с меньшим размером дайджеста.

5.2.3.3.1. Зависящие от алгоритма поля подписей Ed25519Legacy (устарели)

Два значения MPI для Ed25519Legacy представляют строки октетов R и S алгоритма цифровой подписи с кривой Эдвардса (Edwards-curve Digital Signature Algorithm или EdDSA) [RFC8032].

- MPI точки R в виде (беспрефиксной) естественной (little-endian) строки октетов (до 32).
- MPI значения EdDSA S в (беспрефиксном) естественном (little-endian) формате размером до 32 октетов.

Ed25519Legacy **недопустимо** применять в пакетах Signature версии 6 и выше.

5.2.3.4. Зависящие от алгоритма поля подписей Ed25519

- 64 октета естественной подписи.

Дополнительные сведения приведены в параграфе 12.7.

Подписи версии 3 **недопустимо** создавать и использовать с Ed25519.

[1] В оригинале этот пункт отличался, см. https://www.rfc-editor.org/info/rfc9580/. *Прим. перев.*

В подписях Ed25519 **должен** применяться алгоритм хэширования с размером дайджеста не менее 256 битов. Проверяющая реализация **должна** отвергать подписи Ed25519, использующие алгоритм хэширования с меньшим размером дайджеста.

5.2.3.5. Зависящие от алгоритма поля подписей Ed448

- 114 октетов естественной подписи.

Дополнительные сведения приведены в параграфе 12.7.

Подписи версии 3 **недопустимо** создавать и использовать с Ed448.

В подписях Ed448 **должен** применяться алгоритм хэширования с размером дайджеста не менее 512 битов. Проверяющая реализация **должна** отвергать подписи Ed448, использующие алгоритм хэширования с меньшим размером дайджеста.

5.2.3.6. Замечания по подписям

Хэшируется конкатенация подписываемых данных, данных подписи от номера версии до хэшированных данных субпакета (включительно) и (для подписей версии выше 3) 6-октетного трейлера (параграф 5.2.4). Полученное в результате значение является подписью. Старшие 16 битов (2 первых октета) хэш-значения включаются в пакет Signature, чтобы обеспечить возможность отклонения непригодных подписей без проверки подписи целиком. При проверке подписи версии 6 реализация **должна** отвергать подпись, если эти два октета не совпадают с двумя первыми октетами рассчитанного хэш-значения.

Субпакеты Signature состоят из двух полей. Первое поле хэшируется вместе с остальными данными подписи, а второе не хэшируется в подпись. Другой набор субпакетов (нехэшированные) не имеет криптографической защиты с помощью подписи и в него следует включать лишь рекомендательные сведения (параграф 13.13).

Между пакетами подписи версий 4 и 6 имеется два различия. Во-первых, в пакетах версии 6 увеличены размеры полей, указывающих размер хэшированных и нехэшированных субпакетов, что позволяет включать в субпакеты значительно больше данных. Во-вторых, при хэшировании применяется случайная затравка (salt, параграф 13.2).

Алгоритмы преобразования результата хэш-функции в подпись описаны в параграфе 5.2.4.

5.2.3.7. Спецификация субпакета подписи

Набор данных субпакетов может включать субпакеты Signature. В пакетах Signature данным субпакетов предшествует 2-х (версия 4) или 4-октетный (версия 6) скалярный счётчик числа октетов во всех субпакетах. Увеличение указателя на это значение позволяет пропустить все данные субпакетов.

Каждый субпакет состоит из заголовка и тела. Заголовок состоит из:

- кодированного размера субпакета (1, 2 или 5 октетов);
- кодированного Subpacket Type ID (1 октет);
- зависящих от типа субпакета данных.

Поле размера субпакета учитывает закодированный Subpacket Type ID и зависящие от типа субпакета данные, но не учитывает само поле размера. Размер кодируется аналогично 1-, 2- или 5-октетному заголовку пакета OpenPGP. Декодирование размера выполняется, как показано ниже:

```
if первый октет <  192, then
    lengthOfLength = 1
    subpacketLen = 1st_octet

if первый октет >= 192 и < 255, then
    lengthOfLength = 2
    subpacketLen = ((1st_octet - 192) << 8) + (2nd_octet) + 192

if первый октет = 255, then
    lengthOfLength = 5
    размер субпакета = [4-октетный скаляр, начиная со второго октета]
```

Бит 7 кодированного Subpacket Type ID является битом критичности (critical) и его установка показывает, что субпакет является критичным при распознавании подписи. Если помеченный как критичный субпакет не известен реализации, оценивающему **следует** считать подпись ошибочной. Реализациям **следует** игнорировать некритичные субпакеты нераспознанного типа.

Оценивающая сторона может «распознать» субпакет, но не поддерживать его. Назначение бита критичности состоит в том, чтобы позволить подписывающей стороне сказать оценивающему, что она предпочла бы сообщение об ошибке вместо игнорирования новой, неизвестной функции.

Остальные биты кодированного Subpacket Type ID (биты 6-0) содержат идентификатор типа субпакета.

В таблице 5 приведены определённые в настоящее время субпакеты подписи.

Таблица 5. Реестр типов субпакетов подписей OpenPGP.

ID	*Описание*	*Документ*
0	Резерв	
1	Резерв	
2	Время создания подписи	параграф 5.2.3.11
3	Время завершения действия подписи	параграф 5.2.3.18
4	Экспортируемая подпись	параграф 5.2.3.19
5	Доверительная подпись	параграф 5.2.3.21
6	Регулярное выражение	параграф 5.2.3.22
7	Возможен отзыв	параграф 5.2.3.20

8	Резерв	
9	Время завершения действия ключа	параграф 5.2.3.13
10	Заглушка для совместимости с прежними версиями	
11	Предпочтительные симметричные шифры для v1 SEIPD	параграф 5.2.3.14
12	Ключ отзыва (устарел)	параграф 5.2.3.23
13-15	Резерв	
16	Идентификатор ключа эмитента	параграф 5.2.3.12
17-19	Резерв	
20	Данные обозначения	параграф 5.2.3.24
21	Предпочтительные алгоритмы хэширования	параграф 5.2.3.16
22	Предпочтительные алгоритмы сжатия	параграф 5.2.3.17
23	Предпочтения сервера ключей	параграф 5.2.3.25
24	Предпочтительный сервер ключей	параграф 5.2.3.26
25	Первичный идентификатор пользователя	параграф 5.2.3.27
26	URI политики	параграф 5.2.3.28
27	Флаги ключа	параграф 5.2.3.29
28	Идентификатор подписавшего пользователя	параграф 5.2.3.30
29	Причина отзыва	параграф 5.2.3.31
30	Свойства (функции)	параграф 5.2.3.32
31	Назначение подписи	параграф 5.2.3.33
32	Вложенная подпись	параграф 5.2.3.34
33	Оттиск эмитента	параграф 5.2.3.35
34	Резерв	
35	Оттиск предполагаемого получателя	параграф 5.2.3.36
37	Резерв (аттестованные сертификаты)	
38	Резерв (блок ключей)	
39	Предпочтительные шифры AEAD	параграф 5.2.3.15
100-110	Приветное использование и эксперименты	

Реализациям **следует** поддерживать 4 субпакета предпочтительных алгоритмов (11, 21, 22, 39), а также «Свойства» (30) и «Причину отзыва» (29). Для предотвращения скрытой пересылки (параграф 13.12) реализациям следует поддерживать субпакеты «Оттиск предполагаемого получателя» (35). Если реализация отказывается от от поддержки некоторых пакетов предпочтений, она **должна** по умолчанию поддерживать обязательные для реализации алгоритмы, чтобы обеспечить функциональную совместимость. Реализациям шифрования, не поддерживающим субпакеты «Свойства» (30), следует выбрать тип формата шифрованных данных на основе версий ключей получателя или внешних допущений (см. параграф 13.7).

5.2.3.8. Типы субпакетов подписи

В настоящее время для подписей OpenPGP определено множество субпакетов, часть которых относится к самой подписи, а некоторые являются атрибутами ключа. Субпакеты, встречающиеся в самоподписи, помещаются в сертификат, созданный самим ключом. Отметим, что у ключа может быть более одного User ID и, следовательно, более одной самоподписи и несколько субпакетов.

Субпакеты могут размещаться в хэшированной или нехэшированной части подписи. Если субпакет не хэширован, информацию в нем нельзя считать окончательной (определённой), поскольку она не учтена в криптографической подписи. Дополнительные сведения о хэшированных и нехэшированных субпакетах приведены в параграфе 13.13.

5.2.3.9. Замечания по субпакетам

Возможны противоречия между сведениями из разных субпакетов подписи, например, несколько уровней предпочтения или сроков действия. В большинстве случаев реализации следует использовать последний субпакет из хэшированного раздела, но можно использовать более осмысленную схему разрешения конфликтов. Выбор схемы разрешения конфликтов намеренно оставлен разработчикам, поскольку большинство конфликтов - это просто синтаксические ошибки, а неоднозначная формулировка позволяет получателю быть снисходительным, а отправителю - строгим при генерации.

Некоторые очевидные конфликты могут иметь значение на практике. Предположим, например, что владелец имеет ключи версий 3 и 4 с общим ключевым материалом RSA. Любой из этих ключей позволяет проверить созданную другим подпись и убедиться, что она содержит субпакет Issuer Key ID (параграф 5.2.3.12) для каждого ключа как способ явной привязки этих ключей к подписи.

5.2.3.10. Замечанию по самоподписыванию

Самоподписью считается привязывающая подпись, созданная с ключом, на который подпись ссылается. Имеется 3 типа самоподписей: подпись сертификата (Type ID 0x10-0x13), подпись Direct Key (Type ID 0x1F) и подпись привязки субключа (Type ID 0x18). Криптографически достоверные самоподписи следует воспринимать от любого первичного ключа, независимо от применяемых к нему флагов (параграф 5.2.3.29). В частности, первичный ключ не обязан иметь установленный бит 0x01 в первом октете флагов ключа для того, чтобы самоподпись была действительной.

Для самоподписанных сертификатов каждый User ID **может** иметь самоподпись и, соответственно, разные субпакеты в этих самоподписях. Для подписей Subkey Binding каждый субключ **должен** иметь самоподпись. Субпакеты из самоподписи сертификата применяются к User ID, а субпакеты из самоподписи субключа - к субключу. Субпакеты подписи Direct Key применяются ко всему ключу.

Реализации следует интерпретировать субпакеты предпочтений самоподписи как можно более узко (точно). Предположим, например, что ключ имеет два имени пользователей - Alice и Bob. Предположим, что Алиса предпочитает шифр AEAD AES-256 с OCB, а Боб - Camellia-256 с GCM. Если реализация находит ключ по имени Алисы, предпочтительным шифром AEAD будет AES-256 с OCB, если же ключ найден по имени Боб, предпочтительным будет алгоритм Camellia-256 с GCM. Если ключ найден по Key ID, алгоритм Primary User ID для ключа обеспечивает выбор предпочтительного шифра AEAD.

Отзыв самоподписи или разрешение на завершение срока её действия зависит от типа подписи. Отзыв самоподписи User ID делает имя пользователя недействительным. Самоподпись является утверждением: «Моё имя X связано с моим ключом K» и подтверждается сертификатами других пользователей. Если другой пользователь отзывает свою сертификацию, он фактически сообщает, что больше не доверяет привязке этого имени к ключу. Если сам пользователь отзывает свою самоподпись, он больше не применяет это имя, не имеет соответствующего адреса электронной почты и т. п. Отзыв связующей подписи фактически уничтожает субключ, отзыв подписи Direct Key отменяет подпись. Более подробные сведения приведены в параграфе 5.2.3.31.

Поскольку самоподпись содержит важные сведения об использовании ключа, реализациям **следует** разрешать пользователю переписывать самоподпись и важную информацию в ней, такую как предпочтения и срок действия ключа.

Когда реализация импортирует секретный ключ, ей **следует** убедиться в том, что внутренние самоподписи ключа не анонсируют свойства или алгоритмы, которые реализация не поддерживает. При обнаружении такого несоответствия реализации **следует** предупредить пользователя и предложить создать новые самоподписи, анонсирующие свойства и алгоритмы, которые реализация поддерживает.

Реализация, столкнувшаяся с несколькими самоподписями одного объекта, **должна** выбрать из них наиболее свежую действительную подпись, игнорируя остальные.

По соглашению ключ версии 4 включает сведения о первичном открытом ключе (флаги, срок действия и т. п.) и переносимом открытом ключе в целом (свойства, предпочтения алгоритмов и т. п.) в самоподписи User ID типа 0x10 или 0x13. Для использования ключа версии 4 в некоторых реализациях требуется наличие хотя бы одного User ID с действительной самоподписью. По этой причине **рекомендуется** включать в такие ключи по меньшей мере один идентификатор пользователя с самоподписью.

Для ключей версии 6 **рекомендуется** хранить сведения о первичном открытом ключе, а также переносимом открытом ключе в целом (флаги, срок действия, свойства, предпочтения алгоритмов и т. п.) в подписи Direct Key (тип 0x1F) для Public Key вместо размещения в самоподписи User ID. Реализация **должна** убедиться в наличии подписи Direct Key перед использованием ключа версии 6. Это предотвратит некоторые атаки, в которых злоумышленник вырезает самоподпись, задающую время завершения срока действия ключа (Key Expiration Time) или некоторые предпочтения.

Реализациям **не следует** требовать наличия самоподписи User ID для восприятия и использования ключа, если только конкретное применение не зависит от самоидентификации владельца ключа текстовой меткой в User ID. Например, при обновлении ключа для получения сведений об изменении срока действия, анонсируемых свойствах, предпочтениях алгоритмов, отзыве, ротации субключей и т. п. не нужно требовать самоподпись User ID. С другой стороны, при проверке подписи в сообщении электронной почты реализация **может** выбрать восприятие лишь подписи с ключом, имеющим действительную самоподпись User ID, соответствующего заголовку From: в сообщении, чтобы избежать атак с подменой подписи.

5.2.3.11. Время создания подписи

(4-октетное поле времени)

Время создания подписи.

Этот субпакет **должен** присутствовать в хэшированной области. При генерации пакет **следует** помечать как критический.

5.2.3.12. Идентификатор ключа эмитента

(8 октетов Key ID)

OpenPGP Key ID выдавшего подпись ключа. Если версия ключа больше 4, этот субпакет **недопустимо** включать в подпись и взамен следует использовать субпакет Issuer Fingerprint (параграф 5.2.3.35). В прежних версиях спецификации этот субпакет назывался Issuer.

5.2.3.13. Время завершения действия ключа

(4-октетное поле времени)

Срок действия ключа, указываемый числом секунд с момента создания ключа до момента завершения его действия. Для прямой и сертификационной самоподписи временем создания ключа является время создания первичного ключа, для подписи привязки субключа - время создания субключа. Отсутствие или нулевое значение параметра означает неограниченный срок действия ключа. Субпакет применяется лишь в самоподписи. При генерации субпакет **следует** помечать как критический.

5.2.3.14. Предпочтительные симметричные шифры для SEIPD v1

(массив 1-октетных значений)

Последовательность Symmetric Cipher Algorithm ID, указывающих предпочтения владельца ключа в части получения пакетов данных версии 1 с шифрованием и защитой целостности (параграф 5.13.1). Тело субпакета представляет собой упорядоченный по снижению предпочтения список октетов. Предполагается, что реализация получателя поддерживает только перечисленные алгоритмы. Значения Algorithm ID указаны в параграфе 9.3. Субпакет применяется лишь в самоподписи. При генерации пакета SEIPD v2 список предпочтений не применим (параграф 5.2.3.15).

5.2.3.15. Предпочтительные шифры AEAD

(массив пар октетов, указывающих алгоритмы симметричного шифрования и AEAD)

Последовательность парных идентификаторов алгоритмов, указывающих предпочтения владельца ключа в части получения пакетов данных версии 1 с шифрованием и защитой целостности (параграф 5.13.2). Каждая пара октетов указывает комбинацию симметричного шифра и режима AEAD, которые предпочитает владелец ключа. В каждой паре указывается сначала Symmetric Cipher Algorithm ID, затем идентификатор алгоритма AEAD. Тело субпакета представляет собой упорядоченный по снижению предпочтения список пар октетов. Предполагается, что реализация

получателя поддерживает только перечисленные комбинации в дополнение к обязательным для реализации AES-128 и OCB. Если AES-128 и OCB не указаны в субпакете, они предполагаются размещёнными в конце списка. Идентификаторы алгоритмов AEAD приведены в параграфе 9.6, Symmetric Cipher Algorithm ID - в параграфе 9.3.

Например, пакет с 6 октетами

```
09 02 09 03 13 02
```

указывает, что владелец ключа предпочитает получать SEIPD v2 с использованием AES-256 с OCB, затем AES-256 с GCM, затем Camellia-256 и OCB, а также не указанный явно AES-128 с OCB.

Отметим, что поддержка пакетов версии 2 с симметричным шифрованием и защитой целостности (параграф 5.13.2) в общем случае указывается флагом Features (параграф 5.2.3.32). Субпакет применяется лишь в самоподписи. При генерации пакета SEIPD v1 список предпочтений не применим (параграф 5.2.3.14).

5.2.3.16. Предпочтительные алгоритмы хэширования

(массив 1-октетных значений)

Массив идентификаторов дайджеста сообщения, который указывает предпочтительные для владельца ключа алгоритмы. Список упорядочен, подобно списку предпочтительных шифров AEAD, идентификаторы алгоритмов заданы в параграфе 9.5. Субпакет применяется лишь в самоподписи.

5.2.3.17. Предпочтительные алгоритмы сжатия

(массив 1-октетных значений)

Идентификаторы алгоритмов сжатия, указывающие предпочтительные для владельца ключа алгоритмы. Список упорядочен, подобно списку шифров AEAD, идентификаторы алгоритмов заданы в параграфе 9.4. Нулевое значение или отсутствие субпакета обозначают предпочтительность несжатых данных (реализация у владельца ключа может не поддерживать сжатие). Субпакет применяется лишь в самоподписи.

5.2.3.18. Время завершения действия подписи

(4-октетное поле времени)

Срок действия подписи, указанный числом секунд с момента создания подписи, когда срок действия подписи истекает. Отсутствие или нулевое значения параметра означает неограниченный срок действия ключа. При генерации субпакет **следует** помечать как критический.

5.2.3.19. Экспортируемая сертификация

(1 октет, 0 - нет, 1 - да)

Этот субпакет указывет, является ли подпись «экспортируемой», т. е. предназначенной не только для её эмитента. Тело пакета содержит флаг, указывающий, является ли подпись экспортируемой. В случае отсутствия субпакета подпись считается экспортируемой (эквивалентно значению флага 1). Неэкспортируемыми или «локальными» являются подписи, созданные пользователем для обозначения пригодности ключа лишь в рамках пользовательской реализации. Таким образом, при подготовке реализацией пользовательской копии ключа для передачи другому пользователю («экспорт» ключа) из него удаляются локальные подписи.

Получатель переносимого ключа «импортирует» его и аналогичным способом вырезает все локальные сертификаты. При нормальной работе локальных сертификатов не будет, поскольку импорт выполняется для экспортируемого ключа. Однако в некоторых случаях локальные сертификаты возможны. Например, если реализация позволяет импортировать ключи из базы данных в дополнение к экспортируемому ключу, может возникнуть такая ситуация.

Некоторые реализации не представляют интересы лишь одного пользователя (например, сервер ключей). Они всегда вырезают локальные сертификаты из обрабатываемы ключей.

Если реализация генерирует такой субпакет с неэкспортируемой подписью, он **должен** помечаться как критичный.

5.2.3.20. Возможность отзыва подписи

(1 октет, 0 - безотзывная, 1 - отзываемая)

Статус возможности отзыва подписи. Тело пакета содержит флаг, указывающий, является ли подпись отзываемой. Для безотзывных подписей последующие Revocation Signature игнорируются. Такие подписи указывают обязательство подписавшего в части невозможности отзыва его подписи в течение срока действия ключа. При отсутствии субпакета подпись считается отзываемой.

5.2.3.21. Подпись доверия

(1 октет «уровня» (глубины) и 1 октет степени доверия)

Подписавший утверждает, что ключ не только действителен, но и обладает указанным уровнем доверия. Уровень 0 соответствует обычной подписи достоверности, 1 указывает, что подписанный ключ утверждается как действительный доверенный представитель, а октет 2 в теле указывает степень доверия. Уровень 2 говорит, что подписанному ключу доверяют выдачу подписей доверия уровня 1, т. е. ключ является «метапредставителем» (meta introducer). В общем случае подпись доверия уровня n говорит, что ключ является доверенным для выдачи подписей доверия уровня n-1. Степень доверия указывается числом от 0 до255, при этом значения меньше 120 означают частичное доверие, а остальные - полное. Реализациям **следует** выдавать значение 60 для частичного доверия и 120 - для полного.

5.2.3.22. Регулярное выражение

(регулярное выражение с null-символом в конце, кодировка UTF-8)

Применяется вместе с пакетами Signature (level > 0) для ограничения расширения области доверия. Расширение доверия, заданное субпакетом Trust Signature, применимо лишь к подписям для User ID, соответствующих регулярному выражению в теле этого пакета. В регулярных выражениях применяется синтаксис Henry Spencer [REGEX], описанный

в разделе 8. Регулярное выражение сопоставляется с последовательностью символов Unicode UTF-8 из User ID. Само регулярное выражение также использует только символы UTF-8.

По историческим причинам эти субпакеты включают null-символ (октет 0) после регулярного выражения. При разборе реализацией субпакета Regular Expression она **должна** удалять этот октет, а при его отсутствии **должна** отвергать критические субпакеты и игнорировать некритические. При создании субпакета Regular Expression он **должен** включать null-символ. При генерации субпакет **следует** помечать как критический.

5.2.3.23. Ключ отзыва (отменён)

(1 октет класса, 1 октет идентификатора открытого ключа, 20 октетов оттиска версии 4)

Этот механизм устарел и приложениям **недопустимо** создавать такие субпакеты. Приложения, которым нужны отзывы делегирования, могут использовать депонированную Revocation Signature (параграф 13.9). Далее описано, как некоторые реализации пытаются интерпретировать эти устаревшие субпакеты.

Пакет предназначен для предоставления указанному ключу права выдачи подписей отзыва для данного ключа. В октете класса должен быть установлен (1) бит 0x80. Установленный бит 0x40 указывает, что сведения об отзыве являются деликатными (sensitive). Остальные биты предназначены для будущих расширений на другие варианты полномочий. Параметр применяется только в самоподписях Direct Key (Type ID 0x1F), а использование в других типах самоподписей не задано.

Если установлен флаг sensitive, владелец ключа считает, что субпакет содержит приватные сведения о доверии, которые описывают деликатные взаимоотношения в реальном мире. При установленном флаге реализациям **не следует** экспортировать такую подпись другим пользователям, за исключением случаев, когда данные должны быть доступны, т. е. подпись отправляется назначенному отзывающему (revoker) или сопровождается Revocation Signature от него. Отметим, что может быть разумно вынести этот субпакет в отдельную подпись, чтобы он не сочетался с другими субпакетами, которые нужно экспортировать.

5.2.3.24. Данные обозначения

(4 октета флагов, 2 октета размера имени (M), 2 октета размера значения (N), M октетов имени, N октетов значения)

Этот субпакет описывает «обозначение» (notation) подписи, которое хочет сделать эмитент. Обозначение имеет имя и значение, являющиеся строками октетов. В подписи может быть несколько обозначений. Обозначения могут применяться для любых расширений, которые хочет внести эмитент подписи. Поле flags содержит 4 октета флагов. Не определённые флаги **должны** быть сброшены (0). Определённый в настоящее время флаг указан в таблице 6.

Таблица 6. Реестр флагов субпакетов OpenPGP Signature Notation Data.

Позиция флага	***Сокращение***	***Описание***
0x80000000 (первый бит первого октета)	human-readable	Текст UTF-8

Имена обозначений являются произвольными строками в кодировке UTF-8. Они относятся к двум пространствам имён - IETF и пользовательскому. Имена в пространстве IETF регистрируются IANA и в них **недопустимо** использовать символ @ (0x40), который служит тегом пользовательского пространства имён.

Таблица 7. Реестр типов субпакетов OpenPGP Signature Notation Data.

Имя	***Тип данных***	***Разрешённые значения***
Регистраций пока нет.		

Имена в пользовательском пространстве состоят из строки UTF-8, за которой следует символ @ и доменное имя DNS. Отметим, что в строку тега **недопустимо** включать символ @. Например, корпорация Example может задать имя sample@example.com. Имена в пользовательском пространстве принадлежат владельцу домена и контролируются им. В общем случае считается некорректным создание имени в чужом пространстве DNS.

Поскольку имена в пользовательском пространстве имеют форму адресов электронной почты, при внедрении **может** возникнуть желание связать это имя с человеком, с которым можно связаться по поводу использования именованного тега. Отметим, что использование кодировки UTF-8 ведёт к тому, что не все корректные имена из пользовательского пространства являются допустимыми адресами электронной почты.

Если обозначение является критическим, это относится именно к нему, а не к обозначениям в общем смысле.

5.2.3.25. Предпочтения для сервера ключей

(N октетов флагов)

Флаги предпочтений владельца ключа в части обработки ключа на сервере ключей. Не заданные флаги **должны** быть сброшены (0).

Таблица 8. Реестр флагов предпочтения серверов ключей OpenPGP.

Флаг	***Сокращение***	***Определение***
0x80...	No-modify	Владелец ключа просит, чтобы ключ мог обновить или изменить только он или администратор сервера.

Применяется только в самоподписях.

5.2.3.26. Предпочтительный сервер ключей

(строка)

URI сервера ключей, который владелец ключа предпочитает использовать для обновлений. Ключи с несколькими User ID могут иметь Preferred Key Server для каждого User ID. Поскольку указывается URI, сервер ключей фактически может быть копией, полученной по https, ftp, http и т. п.

5.2.3.27. Первичный идентификатор пользователя

(1 октет, Boolean)

Флаг в самоподписи User ID, указывающий, является ли этот идентификатор первичным User ID для данного ключа. Реализация может устранять неоднозначности в предпочтениях, например, обращаясь в первичному User ID. При отсутствии флага предполагается значение 0. Если в качестве основного для ключа указано несколько User ID, реализация может устранить неоднозначность любым способом, но **рекомендуется** отдавать предпочтение User ID с наиболее свежей самоподписью.

При наличии в самоподписи пакета User ID субпакет относится только к этому пакету User ID, в случае пакета User Attribute - только к пакету User Attribute. Таким образом, существует два независимых «основных» идентификатора - один для User ID, другой для User Attribute.

5.2.3.28. URI политики

(строка)

Этот субпакет содержит URI документа, описывающего правила, по которым была выпущена подпись.

5.2.3.29. Флаги ключа

(N октетов флагов)

Этот субпакет содержит набор флагов с информацией о ключе. Флаги являются строкой октетов и реализациям **недопустимо** предполагать для неё фиксированный размер, поскольку тот может меняться с течением времени. Если список короче ожидаемого реализацией, отсутствующие флаги предполагаются нулевыми. Определения флагов даны в таблице 9.

Таблица 9. Реестр флагов ключей OpenPGP.

Флаг	***Определение***
0x01...	Ключ может служить для сертификации User ID (Signature Type ID 0x10-0x13) или подписей Direct Key (Signature Type ID 0x1F) над другими ключами.
0x02...	Ключ может служить для подписывания данных.
0x04...	Ключ может служить для шифрованных коммуникаций.
0x08...	Ключ может служить для хранилища с шифрованием.
0x10...	Секретная часть ключа может быть расщеплена с помощью механизма обобществления ключа.
0x20...	Ключ может служить для аутентификации.
0x80...	Секретная часть этого ключа может находиться у нескольких лиц.
0x0004...	Резерв (ADSK).
0x0008...	Резерв (временные метки).

Флаги этого пакета могут встречаться в самоподписях и удостоверяющих подписях. Их назначение зависит от того, кто делает заявление. Например, удостоверяющая подпись с флагом подписывания данных указывает, что сертификат предназначен для подписи. Флаг шифрованных коммуникаций в самоподписи указывает предпочтение использования данного ключа в коммуникациях. Однако чёткое разделение между коммуникациями и хранением является сложным и полностью зависит от реализации, а авторы документа не претендуют на особую мудрость в этом вопросе, понимая, что общепринятое мнение может измениться.

Флаги расщеплённого (0x10) и группового (0x80) ключа помещаются только в самоподписи и не имеют смысла в подписях сертификатов. Их **следует** включать лишь в подписи Direct Key (Type ID 0x1F) или Subkey Binding (Type ID 0x18) со ссылкой на ключ, к которому применяется подпись.

При генерации субпакет **следует** помечать как критический.

5.2.3.30. User ID подписавшего

(строка)

Этот субпакет позволяет владельцу ключа указать, какой из User ID отвечает за подпись. Многие владельцы используют один ключ для разных целей, таких деловые и личные взаимодействия. Этот субпакет позволяет владельцу указать роль подписи. Субпакет не подходит для использования в качестве ссылки на пакет User Attribute.

5.2.3.31. Причина отзыва

(1 октет кода отзыва, N октетов строки указания причины)

Субпакет применяется только в подписях Key Revocation и Certification Revocation, описывая причину отзыва ключа или сертификата. Первый октет содержит машиночитаемый код, указывающий причину отзыва, как показано в таблице 10.

Таблица 10. Реестр причин отзыва OpenPGP (октет отзыва).

Код	***Причина***
0	Причина не указана (Key Revocation или Certification Revocation)
1	Ключ заменён (Key Revocation)
2	Ключевой материал скомпрометирован (Key Revocation)
3	Ключ отменен и больше не применяется (Key Revocation)
32	Сведения User ID больше не действительны (Certification Revocation)
100-110	Приватное использование.

За кодом отзыва следует строка октетов со сведениями о причине отзыва в понятной человеку форме (UTF-8). Строка может быть пустой (размер 0). Размер субпакета является размером строки причины + 1. Реализациям следует поддерживать эти субпакеты, включать их во все Revocation Signature и подобающим образом интерпретировать. Причины существенно различаются семантически и важно указывать их в подписях отзыва.

При отзыве ключа из-за компрометации, все созданные с его помощью подписи попадают под подозрение. Однако при простой смене или отмене ключа прежние подписи остаются действительными. Если отзывается самоподпись

удостоверения User ID, отзыв говорит о том, что имя пользователя больше не применяется. В такой отзыв **следует** включать субпакет причины с кодом 32. Отметим, что отозвать можно любой сертификат, включая сертификат другого лица. Имеется много веских причин для отзыва удостоверяющей подписи, таких как уход владельца ключа из компании. Отозванный сертификат больше не является действительным.

5.2.3.32. Свойства

(N октетов флагов)

Субпакет Features указывает расширенные возможности OpenPGP, поддерживаемые реализацией. Это сделано для того, чтобы по мере добавления в OpenPGP функций, не совместимых с прежними версиями, пользователь мог заявить возможность их применения. Битовые флаги указывают поддержку соответствующих свойств. Субпакет похож на субпакет предпочтений и применяется только в самоподписях.

Реализации **не следует** применять те или иные свойства при отправке пользователю, который не указал возможность их применения, если она не может узнать об их поддержке с помощью иных механизмов (зависят от реализации).

Определённые в настоящий момент свойства указаны в таблице 11 (первый октет).

Таблица 11. Реестр флагов свойств OpenPGP.

Свойство	***Определение***	***Документ***
0x01...	Пакет с симметричным шифрованием и защитой целостности версии 1.	параграф 5.13.1
0x02...	Резерв.	
0x04...	Резерв.	
0x08...	Пакет с симметричным шифрованием и защитой целостности версии 2.	параграф 5.13.2

При поддержке реализацией какого-либо из указанных свойств ей **следует** поддерживать субпакеты Features. Применение этих субпакетов при генерации шифрованных данных описано в параграфе 13.7.

5.2.3.33. Объект подписи (Target)

(октет алгоритма с открытым ключом, 1 октет алгоритма хэширования, N октетов хэш-значения)

Субпакет указывает конкретный объект, к которому относится подпись. Для Revocation Signature пакет явно указывает отзываемую подпись, для Third-Party Confirmation и Timestamp - подписываемую подпись. Аргументы являются идентификатором целевой подписи. Значение N **должно** совпадать с размером хэша подписи. Например, целевая подпись SHA-1 **должна** иметь 20 октетов хэш-значения.

5.2.3.34. Встроенная подпись

(тело пакета Signature)

Субпакет содержит полное тело пакета Signature, описанного в параграфе 5.2. Это полезно, когда одной подписи подписи нужно сослаться на другую или быть встроенной в неё.

5.2.3.35. Оттиск эмитента

(октет номера версии ключа, N октетов оттиска)

Оттиск ключа OpenPGP, выдавшего подпись. Этот субпакет **следует** включать во все подписи. Если выпускающий ключ имеет версию 4 и субпакет Issuer Key ID (параграф 5.2.3.12) включён в подпись, Key ID в субпакете Issuer Key ID **должен** совпадать с младшими 64 битами оттиска.

Значение N для оттиска ключа версии 4 составляет 20 октетов, для ключа версии 6 - 32. Поскольку версия подписи связана с версией ключа, октет версии в субпакете **должен** совпадать с версией подписи, а при несоответствии принимающая реализаций **должна** считать подпись непригодной (см. параграф 5.2.5).

5.2.3.36. Оттиск предусмотренного получателя

(октет номера версии ключа, N октетов оттиска)

Оттиск ключа OpenPGP для первичного ключа предполагаемого получателя. Если в подпись включён один или несколько таких субпакетов, её **следует** считать действительной только в шифрованном контексте, где ключ, которым зашифрована подпись, является одним из указанных первичных ключей или их субключей. Это может использоваться для предотвращения пересылки подписи за пределы предусмотренного шифрованного контекста (параграф 13.12).

Значение N для оттиска ключа версии 4 составляет 20 октетов, для ключа версии 6 - 32.

Реализации **следует** генерировать такой субпакет при создании подписанного и шифрованного сообщения. При создании субпакета в подписи версии 6 его **следует** помечать как критический.

5.2.4. Расчёт подписей

Все подписи создаются путём хэширования подписываемых данных и последующего применения хэш-значения в алгоритме подписи. При создании подписи версии 6 в контекст хэширования перед данными вносится затравка (salt).

Для подписывания двоичных документов (Type ID 0x00) данные хэшируются напрямую, текстовые документы (Type ID 0x01) реализация **должна** сначала канонизировать, преобразовав завершение строк в <CR><LF> и переведя документ в UTF-8 (см. [RFC3629]). Хэшируется полученный поток байтов UTF-8.

При создании подписи версии 4 для ключа хэш-данные начинаются с октета 0x99, за которым следует 2 октета размера ключа и тело пакета ключа. При создании подписи версии 6 для ключа хэш-данные начинаются с затравки и октета 0x9B, за которым следуют 4 октета размера ключа и тело пакета ключа.

В подписях Subkey Binding (Type ID 0x18) и Primary Key Binding (Type ID 0x19) субключ хэшируется в том же формате, что и основной ключ (с использованием 0x99 или 0x9B в начале). В подписях Primary Key Revocation (Type ID 0x20) хэшируется только отзываемый ключ. В подписи Subkey Revocation (Type ID 0x28) сначала хэшируется основной ключ, затем отзываемый субключ. В подписи Certification (Type ID 0x10 - 0x13) хэшируется User ID, связанный с ключом, в

контексте хэширования после указанных выше данных. В сертификатах версии 3 хэшируется содержимое пакета User ID или User Attribute без заголовка. В сертификатах версий 4 и 6 хэшируется константа 0xB4 (для User ID) или 0xD1 (для User Attribute), за которой следует 4 октета размера данных User ID или User Attribute и сами эти данные. В подписи Third-Party Confirmation (Type ID 0x50) хэшируется затравка (только в версии 6), за которой следует октет 0x88, 4 октета размера подписи и тело пакета Signature (отметим, что это заголовок пакета Legacy для пакета Signature с нулевым значением поля length-of-length). Нехэшированные данные субпакета в пакете Signature не включаются в хэш, а для размера нехэшированных данных субпакета устанавливается значение 0.

После хэширования тела данных хэшируется трейлер, зависящий от версии подписи.

- Для подписи версии 3 хэшируется 5 октетов тела пакета, начиная с поля типа подписи. За этими данными, указывающими тип подписи, следует 4 октета времени создания подписи (Signature Creation Time).
- Для подписи версии 4 или 6 хэшируется тело пакета, начиная с первого поля (номер версии) и заканчивая хэшированными данными субпакета и финальным дополнительным трейлером:
  - октет, указывающий версию подписи (0x04 или 0x06);
  - тип подписи;
  - алгоритм с открытым ключом;
  - алгоритм хэширования;
  - размер хэшируемого субпакета;
  - тело хэшируемого субпакета;
  - второй октет версии (0x04 или 0x06);
  - октет 0xFF;
  - число октетов хэшированных данных от пакета Signature до тела хэшированного субпакета, включительно (4-октетное число без знака в формате по модулю 2^32).

После выполнения хэширования в едином контексте результат хэширования используется в алгоритме подписи, а первые 2 октета помещаются в пакет Signature, как описано в параграфе 5.2.3.

Рабочие примеры данных, хэшированных при подписи, приведены в Приложении A.3.1.

5.2.4.1. Замечания по расчёту подписей

Данные, фактически хэшируемые OpenPGP, меняются в зависимости от версии подписи, чтобы подпись одной версии невозможно было использовать в качестве подписи другой версии для существенно отличающихся данных. Потоки хэшированных данных различаются по трейлерам, в первую очередь в пятом и шестом октете от конца потока:

- в подписях версии 3 пятый от конца октет содержит Signature Type ID, где значение 0xFF **недопустимо**;
- все подписи версий после 3 используют в пятом от конца октете значение 0xFF, а в шестом от конца (перед 0xFF) указывается номер версии подписи.

5.2.5. Неизвестные и некорректно сформированные подписи

Иногда пакет Signature (или соответствующий пакет One-Pass Signature, см. параграф 5.4) может оказаться неизвестным или некорректно сформированным. Некоторые из возможных проблем указаны ниже.

- Неизвестный тип подписи.
- Неизвестная версия подписи.
- Неподдерживаемая версия подписи.
- Неизвестный критический субпакет (см. параграф 5.2.3.7) в хэшированной области.
- Субпакет с отличным от ожидаемого размером.
- Размер хэшированной области субпакета превышает размер самого пакета Signature.
- Заведомо слабый алгоритм хэширования (например, MD5).
- Несоответствие ожидаемого размера затравки алгоритма хэширования фактическому размеру затравки.
- Несовпадение версий One-Pass Signature и Signature (см. параграф 10.3.2.2).
- Подпись версии, отличной от 6, создана с ключом версии 6.

При получении такой подписи реализация **должна** игнорировать её для целей проверки и **недопустимо** сообщать об успешной проверке подписи. В то же время **недопустимо** прерывать обработку потока пакетов или отвергать другие подписи в том же потоке только из-за наличия неизвестной или непригодной подписи. Это требуется для совместимости с прежними версиями. Вывод сообщения об отсутствии успешно проверенных подписей предпочтительней прерывания обработки целиком.

5.3. Пакет ключа симметричного шифрования сеансового ключа (тип 3)

Пакет Symmetric Key Encrypted Session Key (SKESK) содержит ключ симметричного шифрования сеансового ключа, используемого для шифрования сообщения. Контейнеру шифрования (пакетам с симметричным шифрованием и защитой целостности или унаследованным пакетам Symmetrically Encrypted Data), содержащему зашифрованное сообщение, может предшествовать 1 или несколько пакетов Public Key Encrypted Session Key (параграф 5.1) и/или Symmetric Key Encrypted Session Key. Сообщение шифруется с сеансовым ключом, а этот ключ также шифруется и помещается в пакет(ы) Encrypted Session Key. Если контейнеру шифрования предшествует один или несколько пакетов

SKESK, каждый из них задаёт парольную фразу, которая может применяться для расшифровки сообщения. Это позволяет зашифровать сообщение с несколькими открытыми ключами, а также с одной или несколькими парольными фразами.

Тело этого пакета начинается с 1-октетного номера версии типа пакета. В настоящее время определены версии 4 и 6. Оставшаяся часть пакета зависит от версии. Версии различаются способом шифрования сеансового ключа с парольной фразой и кодируемым содержимым. Версия пакета SKESK должна совпадать с версией пакета SEIPD (параграф 10.3.2.1). Новые версии пакетов SKESK следует регистрировать в реестре, заданном в параграфе 10.3.2.1.

5.3.1. Пакет сеансового ключа, зашифрованного симметричным ключом версии 4

Пакет SKESK v4 предшествует пакету SEIPD v1 (см. параграф 5.13.1). В унаследованных данных он иногда встречается перед устаревшим пакетом SED (параграф 5.7). Пакет SKESK v4 **недопустимо** размещать перед пакетом SEIPD v2 (параграф 10.3.2.1).

Пакет сеансового ключа, зашифрованного симметричным ключом версии 4, состоит из:

- 1-октетного номера версии (4);
- 1-октетного идентификатора симметричного алгоритма;
- спецификатора S2K, размер которого зависит от типа (параграф 3.7.1);
- необязательного зашифрованного сеансового ключа, который расшифровывается с помощью объекта S2K.

При отсутствии зашифрованного сеансового ключа (обнаруживается по размеру пакета и спецификатора S2K) алгоритм S2K, применённый к парольной фразе, даёт сеансовый ключ для расшифровки сообщения с использованием Symmetric Cipher Algorithm ID из пакета Symmetric Key Encrypted Session Key. Если зашифрованный ключ присутствует, результат применения алгоритма S2K к парольной фразе используется для расшифровки поля этого ключа с использованием режима CFB и IV из одних нулей. Результат расшифровки состоит из 1-октетного идентификатора, указывающего алгоритм с симметричным ключом, используемый для шифрования последующего шифрованного контейнера, и октетов самого сеансового ключа. Поскольку при расшифровке применяется вектор инициализации (IV), содержащий только 0, спецификатор S2K **должен** использовать значение затравки - Salted S2K, Iterated and Salted S2K или Argon2. Это гарантирует, что ключ расшифровки не будет повторяться даже при совпадении парольной фразы.

5.3.2. Пакет сеансового ключа, зашифрованного симметричным ключом версии 6

Пакет SKESK v6 предшествует пакету SEIPD v2 (параграф 5.13.2). Пакет SKESK v6 **недопустимо** размещать перед пакетом SEIPD v1 или устаревшим пакетом Symmetrically Encrypted Data (параграф 10.3.2.1).

Пакет сеансового ключа, зашифрованного с симметричным ключом версии 6, состоит из:

- 1-октетного номера версии (6);
- 1-октетного скалярного счётчика для 5 последующих полей;
- 1-октетного идентификатора симметричного алгоритма шифрования ID (таблица 21);
- 1-октетного идентификатора алгоритма AEAD (таблица 25);
- 1-октетного скалярного счётчика для следующего поля;
- спецификатора S2K, размер которого зависит от типа (параграф 3.7.1);
- стартового IV, размер которого задаёт алгоритм AEAD;
- зашифрованного сеансового ключа;
- тега аутентификации для режима AEAD.

Ключ шифрования ключа (key-encryption key или KEK) выводится с использованием функции HKDF [RFC5869] с алгоритмом хэширования SHA256 [RFC6234]. Исходный ключевой материал (Initial Keying Material или IKM) для HKDF выводится из S2K. Затравка не применяется. Параметр info состоит из Packet Type ID с кодированием OpenPGP (биты 7 и 6 установлены, биты 5-0 содержат Packet Type ID), версии пакета, алгоритма шифрования (cipher-algo) и режима AEAD (AEAD-mode), используемых для шифрования ключевого материала.

Затем сеансовый ключ шифруется с полученным ключом и использованием алгоритма AEAD, заданного для пакета с симметричным шифрованием и защитой целостности версии 2. Отметим, что блоки (chunk) не применяются и имеется лишь один тег аутентификации. В качестве дополнения указываются Packet Type ID в формате OpenPGP (биты 7 и 6 установлены, биты 5-0 содержат Packet Type ID), номер версии пакета, идентификаторы алгоритмов шифрования и AEAD. Например, дополнительные данные для AES-128 с OCB состоят из октетов 0xC3, 0x06, 0x07, and 0x02.

5.4. Пакет однопроходной подписи (тип 4)

Пакет One-Pass Signature предшествует подписанным данным и содержит сведения, позволяющие получателю начать расчёт хэш-значений, требуемых для проверки подписи. Это позволяет поместить пакет Signature в конец сообщения, чтобы подписывающий мог обработать подписываемое сообщение за один проход. Тело сообщения содержит:

- 1-октетный номер версии (в настоящее время 3 или 6); новые версии пакетов One-Pass Signature следует регистрировать в реестре, описанном в параграфе 10.3.2.2;
- 1 октет Signature Type ID (параграф 5.2.1);
- 1-октетное значение, указывающее применяемый алгоритм хэширования;
- 1-октетное значение, указывающее применяемый алгоритм с открытым ключом;
- в пакетах версии 6 - поле переменного размера, содержащее:

  - 1 октет размера затравки, который **должен** соответствовать алгоритму хэширования (таблица 23);
  - затравка (случайное значение указанного размера), которое **должно** совпадать с полем затравки в соответствующем пакете Signature;
- в пакетах v3 - 8-октетное значение Key ID ключа подписи;
- в пакетах версии 6 - 32 октета оттиска ключа подписи; поскольку подписи версии 6 могут создаваться лишь с ключами версии 6, размер оттиска фиксирован;
- 1-октетное значение флага, указывающее, является ли подпись вложенной (нулевое значение указывает, что следующий пакет является другим пакетом One-Pass Signature (OPS), описывающим другую подпись для применения к тем же данным сообщения).

При создании однопроходной версия пакета OPS **должна** совпадать с версией связанного пакета Signature, за исключением исторической случайности, когда ключи версии 4 используют пакет OPS версии 3 (OPS версии 4 нет). Полное описание соответствия версий Key, Signature и One-Pass Signature приведено в параграфе 10.3.2.2.

Если сообщение содержит не одну подпись OPS, пакеты Signature «заключают сообщение в скобки», т. е. первый пакет Signature после сообщения соответствует последнему пакету One-Pass Signature, а финальный пакет Signature - первому пакету One-Pass Signature.

5.5. Пакеты с ключевым материалом

Пакет с ключевым материалом содержит все сведения о секретном и открытом ключе. Имеется 4 варианта этих пакетов с двумя главными версиями (4 и 6) и двумя настоятельно не рекомендуемыми версиями (2 и 3). Эти варианты описаны ниже. По историческим причинам версии 1 и 5 не были заданы.

5.5.1. Варианты пакетов с ключами

5.5.1.1. Пакет открытого ключа (тип 6)

Пакет Public Key начинает последовательность, формирующую OpenPGP Key (иногда его называют сертификатом OpenPGP).

5.5.1.2. Пакет открытого субключа (тип 14)

Пакет Public Subkey (Type ID 14) имеет такой же формат, как Public Key, но указывает субключ. С одним ключом верхнего уровня может быть связан один или несколько субключей. По традиции, ключ верхнего уровня обеспечивает возможность сертификации, но не предоставляет услуг шифрования, а субключ обеспечивают шифрование (см. параграф 10.1.5).

5.5.1.3. Пакет секретного ключа (тип 5)

Пакет Secret Key содержит все сведения, имеющиеся в пакете Public Key, включая материал открытого ключа, а также секретный ключевой материал после полей открытого ключа.

5.5.1.4. Пакет секретного субключа (тип 7)

Пакет Secret Subkey (Type ID 7) является субключом, аналогичен пакету Secret Key и имеет такой же формат.

5.5.2. Формат пакетов открытых ключей

Имеется 4 версии пакетов ключевого материала. Версии 2 и 3 сочтены устаревшими в 1998 г., версия 4 отменяется этим документом. Реализациям OpenPGP **следует** создавать ключи в формате версии 6. Ключи версии 4 устарели и реализации **не следует** генерировать такие ключи, но **следует** воспринимать их. Ключи версий 3 и 2 **недопустимо** генерировать, но **можно** воспринимать.

Новые версии ключей следует регистрировать в реестре, заданном в параграфе 10.3.2.2.

5.5.2.1. Открытые ключи версии 3

Ключи версии 2 отличаются от ключей версии 3 лишь номером версии. Пакеты Public Key и Public Subkey содержат:

- 1-октетный номер версии (3);
- 4-октетное значение, указывающее время создания ключа;
- 2-октетное значение срока действия ключа в сутках; нулевое значение указывает неограниченный срок;
- 1-октетный идентификатор алгоритма с открытым ключом;
- последовательность MPI, задающих ключевой материал:
  - MPI открытого модуля RSA n;
  - MPI открытого показателя степени для шифрования RSA e.

Ключи версии 3 устарели. Они имеют три недостатка. Во-первых, относительно легко можно создать ключ версии 3 с тем же Key ID, что и у другого ключа, поскольку Key ID - это просто 64 младших бита открытого модуля. Во-вторых, достаточно велика вероятность конфликта оттисков, поскольку для них хэшируется лишь ключевой материал без учёта размера. В-третьих, алгоритм MD5 имеет недостатки, вынуждающие разработчиков выбирать иные алгоритмы. Дополнительное рассмотрение Key ID и оттисков приведено в параграфе 5.5.4.

5.5.2.2. Открытые ключи версии 4

Формат ключей версии 4 похож на формат версии 3, но не включает срок действия ключа, который перенесён в пакет Signature. Оттиски ключей версии 4 рассчитываются иначе (параграф 5.5.4). Пакет версии 4 содержит:

- 1-октетный номер версии (4);
- 4-октетное значение, указывающее время создания ключа;
- 1-октетный идентификатор алгоритма с открытым ключом;
- зависящая от алгоритма последовательность значений, образующих ключевой материал (параграф 5.5.5).

5.5.2.3. Открытые ключи версии 6

Формат ключей версии 6 похож на формат версии 4, но включает счётчик ключевого материала, помогающий анализировать пакеты Secret Key (расширение формата пакетов Public Key) в случае неизвестного алгоритма. Оттиски ключей версии 6 рассчитываются иначе (см. параграф 5.5.4). Пакет версии 6 содержит:

- 1-октетный номер версии (6);
- 4-октетное значение, указывающее время создания ключа;
- 1-октетный идентификатор алгоритма с открытым ключом;
- 4-октетный скаляр, указывающий размер ключевого материала в следующем поле;
- зависящая от алгоритма последовательность значений, образующих ключевой материал (параграф 5.5.5).

5.5.3. Формат пакетов секретных ключей

Пакеты Secret Key и Secret Subkey содержат все данные пакетов Public Key и Public Subkey, а также зависящие от алгоритма данные секретного ключа (в конце), которые обычно зашифрованы. Содержимое пакетов описано ниже.

- Описанные выше поля пакетов Public Key или Public Subkey.
- 1 октет (октет использования S2K), указывающий, защищён ли секретный ключевой материал паролем и как это сделано. Значение 0 указывает, что данные секретного ключа не зашифрованы, 253 (AEAD), 254 (CFB), 255 (MalleableCFB) указывают наличие последующего спецификатора S2K и других параметров. Остальные значения указывают алгоритм шифрования с симметричным ключом. В пакетах версии 6 **недопустимо** использовать значение 255 (MalleableCFB).
- В пакетах версии 6 с зашифрованным ключевым материалом (предыдущий октет не равен 0) - 1-октетный скалярный счётчик общего размера всех последующих (условно включаемых) полей параметров S2K.
- Включаемые условно поля (по значению поля использования S2K) параметров S2K:
  - при значении 253, 254, 255 - 1-октетный идентификатор алгоритма с симметричным ключом;
  - при значении 253 (AEAD) - 1-октетный идентификатор алгоритма AEAD;
  - для пакетов версии 6 при значении 253 или 254 - 1-октетный счётчик размера следующего поля;
  - при значении 253, 254, 255 - спецификатор S2K, размер которого зависит от типа (параграф 3.7.1);
  - при значении 253 (AEAD) - IV, размер которого задаёт алгоритм AEAD (см. параграф 5.13.2), используемый как nonce для алгоритма AEAD;
  - при значении 254, 255 или идентификаторе симметричного алгоритма (секретные данные шифруются с CFB) - IV того же размера, что и блок шифрования.
- Открытые или шифрованные MPI, образующие данные секретного ключа, зависящие от алгоритма и описанные в параграфе 5.5.5. Если октет использования S2K имеет значение 253 (AEAD), эти данные завершаются тегом аутентификации AEAD. Если октет использования S2K имеет значение 254 (CFB), к открытому тексту в конце добавляется 20-октетный хэш SHA-1 зависящей от алгоритма открытой части и шифруется вместе с ним. Если октет использования S2K равен 255 (MalleableCFB) или имеет другое ненулевое значение (идентификатор алгоритма шифрования с симметричным ключом) к открытому тексту в конце добавляется 2-октетная контрольная сумма (сумма всех октетов по модулю 65536) открытой части и шифруется вместе с ним (устарело и **не следует** применять, см. ниже).
- Для пакетов версии 3 и 4 с нулевым октетом использования S2K - 2-октетная контрольная сумма зависящей от алгоритма части (сумма всех октетов по модулю 65536).

Сводка деталей сохранения зависящих от алгоритма секретов приведена в таблице 2.

Отметим, что в формат пакетов версии 6 добавлены 2 счётчика, помогающие анализировать пакет с неизвестными алгоритмами S2K и шифрования с открытым ключом.

Секретные значения MPI могут шифроваться с использованием парольной фразы. При наличии спецификатора S2K он описывает алгоритм преобразования парольной фразы в ключ, в иных случаях просто применяется хэш MD5 для парольной фразы. Реализация, создающая защищённые паролем пакеты Secret Key, **должна** использовать S2K Specifier, поддерживая простое хэширование для совместимости с прежними версиями (только чтение), но **может** продолжать использование имеющихся секретных ключей в старом формате. Шифр для MPI указывается в пакете Secret Key.

Шифрование и расшифровка секретных данных выполняются с использованием ключа, созданного из парольной фразы и IV в пакете. Если октет использования S2K не равен 253, применяется режим CFB. С ключами версии 3 (только RSA) применяется другой режим. Здесь префикс числа битов MPI (2 первых октета) не шифруется, а шифруются лишь MPI (без префикса). Кроме того, состояние CFB ресинхронизируется в начале каждого нового значения MPI, чтобы граница блока CFB совпадала с началом данных MPI. Для пакетов версии 4 и 6 применяется более простой метод, где шифруются все значения MPI (включая префикс счётчика битов).

Если октет использования S2K имеет значение 253, для разделения ключей выводится ключ KEK с использованием функции HKDF [RFC5869]. В качестве алгоритма хэширования для HKDF применяется SHA256 [RFC6234]. IKM для

HKDF - это ключ, выведенный из S2K без применения затравки (salt). Параметр info состоит из Packet Type ID в формате OpenPGP (установлены биты 7 и 6, а в битах 5-0 содержится Packet Type ID), версии пакета, а также алгоритма шифрования и режима, используемых для шифрования ключевого материала.

Затем зашифрованные значения MPI шифруются ещё раз единым блоком с применением одного из алгоритмов AEAD, указанных для пакета данных с симметричным шифрованием и защитой целостности версии 2. Отметим, что блоки (chunk) не используются и имеется лишь один тег аутентификации. В качестве дополнительных данных алгоритму AEAD передаётся Packet Type ID с кодированием OpenPGP (установлены биты 7 и 6, а в битах 5-0 содержится Packet Type ID), за которым следуют поля пакета Public Key, начиная с номера версии. Например, дополнительные данные, используемые с пакетом Secret Key версии 4 состоят из октетов 0xC5, 0x04, за которым следуют 4 октета времени создания, октет, указывающий алгоритм с открытым ключом, и зависящие от алгоритма параметры открытого ключа. Для пакета Secret Subkey первым октетом будет 0xC7. Для пакета ключа версии 6 вторым октетом будет 0x06, а также будет включаться 4-октетный счётчик размера ключевого материала (см. параграф 5.5.2).

2-октетная контрольная сумма после зависящей от алгоритма части является алгебраической суммой по модулю 65536 открытого текста всех зависящих от алгоритма октетов (включая префикс и данные MPI). В ключах версии 3 контрольная сумма содержится в открытом виде, а в ключах версии 4 шифруется, подобно зависящим от алгоритма данным. Значение суммы служит для проверки корректности парольной фразы. Однако такая сумма признана устаревшей и реализациям **не следует** использовать её. Взамен следует применять хэш SHA-1, обозначенный октетом использования 254. Причиной этого является наличие атак с незаметным изменением секретного ключа. Если октет использования S2K имеет значение 253, контрольная сумма SHA-1 не используется, но применяется тег аутентификации алгоритма AEAD.

При расшифровке секретного ключевого материала с использованием любой из этих схем (с ненулевым октетом использования S2K) полученный поток октетов открытого текста должен быть корректно сформирован. В частности, реализациям **недопустимо** интерпретировать октеты за пределами потока открытого текста как часть какого-либо из развёрнутых объектов MPI. Кроме того, реализация **должна** отвергать любой материал секретного ключа, для которого размер открытого текста не совпадает с размером развёрнутых объектов MPI.

5.5.4. Идентификаторы и оттиски ключей

Каждый ключ OpenPGP имеет оттиск (fingerprint) и Key ID, расчёт значений которых зависит от версии ключа. Размер оттиска ключа зависит от версии, а Key ID (применяется только в пакетах PKESK v3, параграф 5.1.1) всегда имеет размер 64 бита. В таблице 12 указаны оттиски и идентификаторы, описанные в последующих параграфах.

Таблица 12. Реестр идентификаторов и оттисков ключей OpenPGP.

Версия ключа	***Оттиск***	***Размер оттиска в битах***	***Key ID***	***Документ***
3	MD5(MPI без октетов размера)	128	Младшие 64 бита модуля RSA	параграф 5.5.4.1
4	SHA1(нормализованный пакет pubkey)	160	Младшие 64 бита оттиска	параграф 5.5.4.2
6	SHA256(нормализованный пакет pubkey)	256	Старшие 64 бита оттиска	параграф 5.5.4.3

5.5.4.1. Key ID и оттиск версии 3

Для ключей версии 3 идентификатор Key ID (8 октетов) содержит младшие 64 бита открытого модуля ключа RSA. Оттиск ключа версии 3 формируется хэшированием MD5 тела (без 2-октетного размера) значений MPI, формирующих ключевой материал (открытый модуль n, за которым следует показатель e). Ключи версии 3 и MD5 устарели.

5.5.4.2. Key ID и оттиск версии 4

Оттиск версии 4 - это 160-битовый хэш SHA-1 для октета 0x99, за которым следует 2-октетный размер пакета и весь пакет Public Key, начиная с поля версии. Key ID - это 64 младших бита оттиска. Ниже приведён пример материала для хэширования с ключом Ed25519.

a.1) 0x99 (1 октет).

a.2) 2-октетный (big-endian) скалярный счётчик октетов (b)-(e).

b) номер версии = 4 (1 октет).

c) метка времени создания ключа (4 октета).

d) алгоритм (1 октет) - 27 = Ed25519 (пример).

e) зависящие от алгоритма поля.

Для Ed25519 (пример)

e.1) 32 октета, представляющие открытый ключ.

5.5.4.3. Key ID и оттиск версии 6

Оттиск версии 6 - это 256-битовый хэш SHA2-256 для октета 0x9B, за которым следует 4-октетный размер пакета и весь пакет Public Key, начиная с поля версии. Key ID - это 64 старших бита оттиска. Ниже приведён пример материала для хэширования с ключом Ed25519.

a.1) 0x9B (1 октет).

a.2) 4-октетный скалярный счётчик октетов (b)-(f).

b) номер версии = 6 (1 октет).

c) метка времени создания ключа (4 октета).

d) алгоритм (1 октет) - 27 = Ed25519 (пример).

e) 4-октетный скалярный счётчик октетов ключевого материала в следующем поле.

f) зависящий от алгоритма ключевой материал.

Для Ed25519 (пример)

f.1) 32 октета, представляющие открытый ключ.

Отметим возможность конфликтов Key ID, т. е. наличия двух ключей с одним Key ID. Вероятность конфликта оттисков значительно ниже, но все равно ненулевая. Если ключи версий 3, 4, 6 используют общий ключевой материал RSA, Key ID и оттиски ключей будут различаться. Key ID и оттиски субключей рассчитываются так же, как для первичных ключей с использованием 0x99 (версия 4) или 0x9B (версия 6) в качестве первого октета (даже если это не будет действительным Packet Type ID для открытого субключа).

5.5.5. Зависящие от алгоритма части ключа

Форматы открытых и секретных ключей определяют зависящие от алгоритма части ключей, как описано ниже.

5.5.5.1. Зависящие от алгоритма части ключа RSA

Для ключей RSA открытый ключ состоит из последовательности многоразрядных целых чисел:

- MPI открытого модуля RSA n,
- MPI открытого показателя степени RSA e.

Секретный ключ состоит из последовательности многоразрядных целых чисел:

- MPI секретного показателя степени RSA d;
- MPI секретного простого числа RSA p;
- MPI секретного простого числа RSA q (p < q);
- MPI u, мультипликативного обращения p по модулю q.

5.5.5.2. Зависящие от алгоритма части ключа DSA

Для ключей DSA открытый ключ состоит из последовательности многоразрядных целых чисел:

- MPI простого числа DSA p;
- MPI порядка группы DSA q (простой делитель p-1);
- MPI генератора группы DSA g;
- MPI значения открытого ключа DSA y (g^x mod p, где x - секрет).

Секретный ключ состоит из одного многоразрядного целого числа:

- MPI секретного показателя степени DSA x.

5.5.5.3. Зависящие от алгоритма части ключа Elgamal

Для ключей Elgamal открытый ключ состоит из последовательности многоразрядных целых чисел:

- MPI простого числа Elgamal p;
- MPI генератора группы Elgamal g;
- MPI значения открытого ключа Elgamal y (g^x mod p, где x - секрет).

Секретный ключ состоит из одного многоразрядного целого числа:

- MPI секретного показателя степени Elgamal x.

5.5.5.4. Зависящие от алгоритма части ключа ECDSA

Для ключей ECDSA открытый ключ состоит из последовательности многоразрядных целых чисел:

- поле переменного размера с OID, состоящее из:
  - 1 октета размера следующего поля (значения 0 и 0xFF зарезервированы для будущих расширений);
  - октетов OID кривой, как указано в параграфе 9.2;
- MPI точки EC, представляющей открытый ключ.

Секретный ключ состоит из одного многоразрядного целого числа:

- MPI целого числа, представляющего секретный ключ (скаляр точки открытой EC).

5.5.5.5. Зависящие от алгоритма части ключа EdDSALegacy (устарело)

Для ключей (устаревших) EdDSALegacy открытый ключ состоит из последовательности многоразрядных целых чисел:

- поле переменного размера с OID кривой, состоящее из:
  - 1 октета размера следующего поля (значения 0 и 0xFF зарезервированы для будущих расширений);
  - октетов OID кривой, как указано в параграфе 9.2;
- MPI точки EC, представляющей открытый ключ Q в естественной форме с префиксом (см. параграф 11.2.2).

Секретный ключ состоит из MPI строки октетов, представляющей естественную форму секретного ключа в зависящем от кривой формате, как описано в параграфе 9.2.1.

Естественной формой секретного ключа EdDSA является последовательность неструктурированных случайных октетов фиксированного размера, соответствующего конкретной кривой. Эта последовательность используется с криптографическим дайджестом для создания зависящего от кривой секретного скаляра и префикса, применяемого при создании подписи. В параграфе 5.1.5 [RFC8032] более подробно описано использование естественных строк октетов для Ed25519Legacy. Значение, хранящееся в пакете OpenPGP EdDSALegacy Secret Key, - это исходная последовательность случайных октетов.

Отметим, что единственной кривой, определённой для использования с EdDSALegacy является Ed25519Legacy OID.

5.5.5.6. Зависящие от алгоритма части ключа ECDH

Для ключей ECDH открытый ключ состоит из последовательности значений:

- поле переменного размера с OID кривой, состоящее из:
  - 1 октета размера следующего поля (значения 0 и 0xFF зарезервированы для будущих расширений);
  - октетов OID кривой, как указано в параграфе 9.2;
- MPI точки EC, представляющей открытый ключ в формате, связанном с кривой, как указано в параграфе 9.2.1;
- поле переменного размера с параметрами функции вывода ключей (key derivation function или KDF):
  - 1 октет размера последующих полей (значения 0 и 0xFF зарезервированы для будущих расширений);
  - 1 октет со значением 1 (резерв для будущих расширений);
  - 1 октет идентификатора хэш-функции, используемой KDF;
  - 1 октет идентификатора симметричного алгоритма, используемого для упаковки (wrap) симметричного ключа, который применяется для шифрования сообщения (параграф 11.5).

Секретный ключ состоит из MPI строки октетов, представляющей секретный ключ в зависящем от кривой формате (параграф 9.2.1).

5.5.5.6.1. Секретный ключевой материал ECDH

При использовании в ECDH кривой NIST P-256, NIST P-384, NIST P-521, brainpoolP256r1, brainpoolP384r1 или brainpoolP512r1 секретные ключи представляются целым числом в стандартной форме MPI. Другие кривые передаются иначе (хотя все равно в виде одного MPI), как описано ниже или в параграфе 9.2.1.

5.5.5.6.1.1. Секретный ключевой материал ECDH Curve25519Legacy (устарело)

Секретные ключи Curve25519Legacy хранятся как стандартное целое число в форме MPI (big-endian) и их **недопустимо** использовать в пакетах ключей версии 6 и выше. Отметим, что порядок октетов в этой форме является обратным по отношению к «естественной» (little-endian) форме, приведённой в [RFC7748].

Целое число секретного ключа Curve25519Legacy для OpenPGP **должно** иметь соответствующую форму, т. е. оно **должно** быть кратным 8 и иметь значение не меньше 2^254 и не больше 2^255. Размер этого MPI в битах по определению равен 255, поэтому два ведущих октета MPI всегда имеют значения 00 FF, а обращение следующих 32 октетов из линии даёт «естественную» форму.

При создании нового секретного ключа Curve25519Legacy из 32 полностью случайных октетов приведённая ниже процедура даёт формат передачи MPI (отметим сходство с decodeScalar25519, как описано в [RFC7748]).

```
def curve25519Legacy_MPI_from_random(octet_list):
    octet_list[0] &= 248
    octet_list[31] &= 127
    octet_list[31] |= 64
    mpi_header = [ 0x00, 0xFF ]
    return mpi_header || reversed(octet_list)
```

5.5.5.7. Зависящая от алгоритма часть ключа X25519

Для ключей X25519 открытый ключ состоит из одного значения - 32 октетов естественного открытого ключа. Секретный ключ также состоит из одного значения - 32 октетов естественного секретного ключа.

Более подробное описание использования естественных строк октетов дано в параграфе 6.1 [RFC7748]. Значение в пакете OpenPGP X25519 Secret Key - это исходная последовательность случайных октетов, в пакете OpenPGP X25519 Public Key - это X25519(secretKey, 9).

5.5.5.8. Зависящая от алгоритма часть ключа X448

Для ключей X448 открытый ключ состоит из одного значения - 56 октетов естественного открытого ключа. Секретный ключ также состоит из одного значения - 56 октетов естественного секретного ключа.

Более подробное описание использования естественных строк октетов дано в параграфе 6.2 [RFC7748]. Значение в пакете OpenPGP X448 Secret Key - это исходная последовательность случайных октетов, в пакете OpenPGP X448 Public Key - это X448(secretKey, 5).

5.5.5.9. Зависящая от алгоритма часть ключа Ed25519

Для ключей Ed25519 открытый ключ состоит из одного значения - 32 октетов естественного открытого ключа. Секретный ключ также состоит из одного значения - 32 октетов естественного секретного ключа.

Более подробное описание использования естественных строк октетов дано в параграфе 5.1.5 [RFC8032]. Значение в пакете OpenPGP Ed25519 Secret Key - это исходная последовательность случайных октетов.

5.5.5.10. Зависящая от алгоритма часть ключа Ed448

Для ключей Ed448 открытый ключ состоит из одного значения - 57 октетов естественного открытого ключа. Секретный ключ также состоит из одного значения - 57 октетов естественного секретного ключа.

Более подробное описание использования естественных строк октетов дано в параграфе 5.2.5 [RFC8032]. Значение в пакете OpenPGP Ed448 Secret Key - это исходная последовательность случайных октетов.

5.6. Пакет сжатых данных (тип 8)

Пакет Compressed Data содержит сжатые данные. Обычно такие пакеты встречаются в содержимом шифрованного пакета или после пакета Signature или One-Pass Signature и содержат пакет Literal Data. Тело пакета состоит из:

- 1 октета, указывающего алгоритм сжатия пакета;
- сжатых данных, образующих оставшуюся часть пакета.

Тело пакета Compressed Data содержит данные, полученные сжатием последовательности пакетов OpenPGP. Дополнительные сведения о формировании сообщений приведены в разделе 10.

Последовательности пакетов с компрессией ZIP имеют форму блоков raw DEFLATE [RFC1951], с компрессией ZLIB - блоков ZLIB [RFC1950], а для последовательностей со сжатием BZip2 применяется алгоритм BZip2 [BZ2].

Реализация, создающая пакеты Compressed Data, **должна** использовать формат кадрирования OpenPGP (параграф 4.2.1) и **недопустимо** генерировать пакеты Compressed Data с форматом Legacy (параграф 4.2.2). Реализация, работающая со старыми данными или устаревшими реализациями без поддержки [RFC2440], **может** интерпретировать пакеты Compressed Data с использованием формата кадрирования Legacy.

5.7. Пакет данных с симметричным шифрованием (тип 9)

Пакет Symmetrically Encrypted Data содержит данные, зашифрованные с помощью алгоритма с симметричным ключом. После расшифровки он содержит другие пакеты (обычно Literal Data или пакеты сжатых данных, но теоретически это может быть иная последовательность пакетов, образующая действительное сообщение OpenPGP).

Эти пакеты устарели и реализациям **недопустимо** создавать их. Такие пакеты **следует** отвергать, прерывая обработку сообщения. Если реализация решает обрабатывать такой пакет, она **должна** выдать чёткое предупреждение о пакете без защиты целостности.

Пакеты этого формата в общем случае невозможно обработать защищённо, поскольку шифротекст в них может быть изменён (malleable). Выбор лучшего контейнера шифрования OpenPGP без этого недостатка рассматривается в параграфе 13.7.

Тело пакета состоит из:

- случайного префикса размером в 1 блок (например, 16 октетов при 128-битовом блоке), за которым следует копия двух последних октетов, зашифрованных в режиме с обратной связью (Cipher Feedback или CFB) и вектором инициализации (IV), состоящим из нулей;
- данных, зашифрованных в режиме CFB, с последними октетами начального шифротекста размером в 1 блок в качестве IV.

Применяемый симметричный шифр может быть указан в предшествующем пакете Public Key или Symmetric Key Encrypted Session Key. В этом случае идентификатор алгоритма шифрования служит префиксом сеансового ключа до его шифрования. Если пакет этого типа не предшествует шифрованным данным, применяется алгоритм IDEA с сеансовым ключом, рассчитанным как хэш MD5 от парольной фразы, хотя такое применение устарело.

Данные шифруются в режиме CFB (параграф 12.9). Для случайного префикса задаётся IV, состоящий из нулей. Вместо рандомизированного шифрования с помощью IV здесь шифруется строка размером в 1 блок шифра плюс 2. Первые октеты размером в 1 блок (например, 16 октетов при 128-битовом блоке) являются случайными, а следующие 2 октета копируются из двух последних октетов первого блока случайных октетов. Например, для 16-октетного блока октет 17 является копией октета 15, а октет 18 - копией октета 16, для шифра с 8-октетным блоком октет 9 является копией октета 7, а октет 10 - копией октета 8 (нумерация октетов с 1). После шифрования этих октетов (размер блока + 2) создаётся новый контекст CFB для шифрования данных, а в качестве IV служит последний октет первого шифротекста размером в 1 блок (эквивалентным вариантом является ресинхронизация состояния CFB - последние октеты шифротекста размером в 1 блок проходят через шифр, а граница блока сбрасывается). Повторение 2 последних октетов в случайном префиксе позволяет получателю незамедлительно проверить корректность сеансового ключа. Советы по использованию «быстрой проверки» приведены в параграфе 13.4.

5.8. Маркер (тип 10)

Тело пакета Marker состоит из 3 октетов 0x50, 0x47, 0x50 (PGP в UTF-8). При получении таких пакетов они **должны** игнорироваться.

5.9. Literal Data (тип 11)

Пакет Literal Data содержит тело сообщения, т. е. данные, не подлежащие дальнейшей интерпретации. Тело пакета описано ниже.

- 1-октетное поле, описывающее форматирование данных. Если это b (0x62), пакет содержит двоичные данные, u (0x75) указывает текст UTF-8, для которого может потребоваться преобразование в локальную кодировку или иное изменение текстового режима. В прежних версиях спецификации OpenPGP для текста использовалось значение t (0x74) без указания кодировки, сейчас **не следует** создавать пакеты с таким значением, а при получении такого пакета **следует** интерпретировать его как текст UTF-8, если достоверный (не подверженный атакам) контекст не указывает иную кодировку. Этот режим устарел и неоднозначен. Некоторые реализации, предшествующие [RFC2440], определяли также значение l как «локальный» режим для местных машинных

преобразований. В [RFC1991] некорректно указано, что этот флаг локального режима имеет значение 1 (ASCII-цифра 1). Оба этих локальных режима признаны устаревшими.

- Имя файла в виде строки (1 октет размера и имя файла), которое может иметь размер 0. Обычно, если источником шифрованных данных является файл, это будет имя зашифрованного файла. Реализация может считать имя файла в пакете Literal Data более полномочным, чем фактическое имя файла.
- 4-октетное число, указывающее дату, связанную с дословными (literal) данными. Обычно это дата изменения файла, время создания пакета или 0, если конкретное время не задано.
- Остальная часть пакета содержит дословные данные. Текст должен использовать кодировку UTF-8 (см. [RFC3629]) и завершения строк <CR><LF> (обычно для сетей). Символы завершения принимающей реализации следует преобразовывать в локально используемые.

Отметим, что подписи OpenPGP не включают октет форматирования, имя файла и поле даты из пакета Literal Data в хэш подписи, поэтому эти поля в подписанном документе не защищены от подделки. Принимающей реализации **недопустимо** считать эти поля криптографически защищёнными окружающей подписью ни при представлении их пользователю, ни при действиях с ними. Из-за присущей этим полям изменчивости, реализациям, генерирующим пакеты Literal Data, **следует** избегать хранения в этих полях значимых сведений. Если реализация считает, что некоторые данные являются текстом UTF-8 (например, если за пакетом Literal Data следует пакет Signature типа 0x01, см. параграф 5.2.1), она **может** установить в октете формата значение u. В иных случаях реализация **должна** устанавливать значение b. **Следует** использовать пустое имя файла (октет размера 0) и нулевое значение метки времени (4 октета нулей).

Приложениям, желающим включить в подпись метаданные файловой системы, рекомендуется подписывать инкапсулированный архив (например, [PAX]).

Генерирующая пакет Literal Data реализация должна использовать для кадрирования формат OpenPGP (параграф 4.2.1) и **недопустимо** создавать пакеты Literal Data с кадрированием Legacy (параграф 4.2.2). Реализации, работающей с унаследованными данными или данными, созданными реализацией, которая ещё не поддерживает [RFC2440], **могут** интерпретировать пакеты Literal Data с форматом кадрирования Legacy.

5.9.1. Специальное имя файла _CONSOLE (устарело)

Для поля имени файла в пакетах Literal Data исторически сложилось специальное имя _CONSOLE, которое говорит о конфиденциальности сообщения (for your eyes only). Принимающей программе следует более аккуратно обрабатывать такие сообщения, например, не сохраняя полученные данные.

Системам OpenPGP, создающим пакеты Literal Data, **недопустимо** зависеть каким-либо способом от соблюдения этой индикации. Указание не может быть выполнено принудительно, а поле не имеет криптографической защиты. В генерируемых пакетах Literal Data **не рекомендуется** применять это специальное имя.

5.10. Доверие (тип 12)

Пакеты Trust применяются только внутри связок (keyring) и обычно не экспортируются. Эти пакеты содержат данные, которые указывают спецификации пользователя для доверенных эмитентов ключей, а также другие сведения, используемые реализацией в части доверия. Формат пакетов Trust определяется реализацией. Эти пакеты **не следует** выдавать в выходные потоки, передаваемые другим пользователям, и **следует** игнорировать их в любом вводе, за исключением локальных файлов связок ключей.

5.11. Идентификатор пользователя (тип 13)

Пакет User ID содержит текст UTF-8, предназначенный для представления имени и адреса электронной почты владельца ключа. По соглашению применяется mailbox, как описано в [RFC2822], но содержимое не ограничивается. Размер пакета в заголовке указывает размер User ID.

5.12. Атрибуты пользователя (тип 17)

Пакет User Attribute является разновидностью пакета User ID. Он может содержать больше типов данных, чем User ID, где разрешён лишь текст. Подобно User ID, пакет User Attribute может быть сертифицирован владельцем ключа (самоподпись) или владельцем другого ключа, желающим сертифицировать пакет. За исключением отмеченного, пакет User Attribute можно использовать везде, где разрешены пакеты User ID.

Хотя пакеты User Attribute не являются обязательными в спецификации OpenPGP, реализациям **следует** обеспечивать по меньшей мере корректную обработку сертификационных подписей в пакетах User Attribute. Простым способом реализации этого является трактовка пакетов User Attribute как User ID с необрабатываемым содержимым, но реализации могут использовать любой желаемый метод.

Пакет User Attribute состоит из одного или нескольких субпакетов атрибутов, каждый из которых содержит заголовок и тело. Заголовок состоит из:

- поля размера (1, 2 или 5 октетов);
- Subpacket Type ID (1 октет).

Далее следуют соответствующие типу субпакета данные. Типы субпакетов указаны в таблице 13.

Таблица 13. Реестр типов субпакетов пользовательский атрибутов OpenPGP.

ID	*Субпакет с атрибутами*	*Документ*
0	Резерв	
1	Субпакет атрибутов изображения	параграф 5.12.1
100-110	Приватное использование и эксперименты	

Реализациям **следует** игнорировать субпакеты неизвестных типов.

5.12.1. Субпакет атрибутов изображения

Субпакет Image Attribute служит для кодирования изображения, предположительно (не обязательно) принадлежащего владельцу ключа. Субпакет начинается с заголовка изображения, два первых октета которого указывают размер заголовка. В отличие от других многооктетных чисел в этом документе здесь используется формат little-endian (по историческим причинам). После размера заголовка следует один октет версии заголовка изображения. В настоящее время определена лишь версия 1, указывающая 16-октетный заголовок изображения. Первые 3 октета заголовка изображения версии 1 имеют значения 0x10, 0x00, 0x01.

Таблица 14. Реестр версий атрибутов изображений OpenPGP.

Версия	***Документ***
1	параграф 5.12.1

Четвёртый октет заголовка изображения версии 1 указывает формат кодирования изображения. В настоящее время определено лишь значение 1 для формата JPEG. Идентификаторы формата 100 - 110 зарезервированы для приватного применения и экспериментов. Оставшаяся часть заголовка изображения версии 1 содержит 12 резервных октетов, которые **должны** иметь значение 0.

Таблица 15. Реестр форматов кодирования атрибутов изображений OpenPGP.

ID	***Кодирование***
0	Резерв
1	JPEG [JFIF]
100-110	Приватное использование и эксперименты

Остальная часть субпакета содержит само изображение. Поскольку в настоящее время задан лишь тип JPEG, изображение кодируется в формате JPEG File Interchange Format (JFIF), который является стандартным для изображений JPEG [JFIF].

Реализация **может** пытаться определить тип изображения, просматривая его данные, если она не способна обработать конкретную версию заголовка изображения или не распознала указанное значение формата кодирования.

5.13. Пакет с симметричным шифрованием и защитой целостности (тип 18)

Пакет SEIPD содержит зашифрованные данные с защитой целостности. После расшифровки пакета он будет содержать другие пакеты, образующие сообщение OpenPGP (параграф 10.3).

Первый октет пакета указывает номер версии, но шифрованные данные разных версий имеют разную структуру. В пакетах версии 1 содержатся данные, зашифрованные алгоритмом с симметричным ключом и защищённые от изменения с помощью алгоритма хэширования SHA-1. Этот механизм представлен в [RFC4880] и обеспечивает некоторую защиту от искажения шифрованных данных. Пакеты версии 2 содержат данные, зашифрованные с помощью конструкции AEAD. Это обеспечивает более строгую криптографически защиту от искажения зашифрованных данных. Более подробные сведения о выборе формате представлены в параграфе 13.7.

Новые версии пакетов SEIPD следует регистрировать в реестре, указанном в параграфе 10.3.2.1.

5.13.1. Пакет с симметричным шифрованием и защитой целостности версии 1

Пакет с симметричным шифрованием и защитой целостности данных версии 1 состоит из:

- 1-октетного номера версии (1);
- зашифрованных данных (вывод выбранной операции с симметричным ключом в режиме CFB).

Применяемый симметричный шифр **должен** быть указан в пакете Public Key или Symmetric Key Encrypted Session Key, предшествующем данному зашифрованному пакету с защитой целостности. В любом случае идентификатор алгоритма шифрования помещается перед сеансовым ключом до его зашифровки.

Данные шифруются в режиме с обратной связью (CFB, см. параграф 12.9) и вектором инициализации (IV), содержащим только 0. Вместо рандомизации шифрования с помощью IV в OpenPGP перед шифрованием данных к ним добавляется префикс в форме строки октетов, размер которой равен размеру блока шифрования плюс 2 октета. Первые октеты группы размером в 1 блок являются случайными, два последних октета - копиями двух предшествующих им октетов. Например, при размере блока 128 битов (16 октетов) префикс будет содержать 16 случайных октетов, затем 2 октета, являющихся копиями 15-го и 16-го октетов, соответственно. В отличие от устаревших пакетов Symmetrically Encrypted Data (параграф 5.7) эти префиксные данные шифруются в том же контексте CFB и ресинхронизация CFB не выполняется. Повторение 16 битов в случайных данных перед сообщением позволяет получателю сразу же проверить корректность сеансового ключа. Советы по корректному применению быстрой проверки приведены в параграфе 13.4.

В конце открытых данных добавляются октеты со значениями 0xD3 и 0x14. Затем открытые данные проходят хэширование SHA-1. На вход функции хэширования поступают данные описанного выше префикса и все открытые данные, включая трейлерные октеты 0xD3 и 0x14. 20 октетов хэш-значения SHA-1 добавляются в конец открытых данных (после октетов 0xD3, 0x14) и результат шифруется в том же контексте CFB. Добавляемую в конце контрольную сумму называют кодом детектирования изменений (Modification Detection Code или MDC).

При расшифровке открытые данные следует хэшировать с помощью SHA-1, включая префикс и трейлерные октеты 0xD3, 0x14, но без последних 20 октетов, содержащих хэш SHA-1. Рассчитанный хэш SHA-1 сравнивается с последними 20 октетами расшифрованных данных. Несоответствие значений говорит об изменении сообщения и **должно** считаться проблемой безопасности. Об этом следует уведомлять пользователя.

Ненормативное пояснение

> Система MDC - механизм защиты целостности пакетов с симметричным шифрованием и защитой целостности версии 1 - была создана для обеспечения средств защиты, менее строгих, чем подпись, но более сильных, нежели простое шифрование CFB. У шифрования CFB имеется ограничение, связанное с тем, что при повреждении шифрованных данных повреждаются не только затронутые, но и следующие блоки. Кроме того, удаление данных из конца шифрованного CFB блока невозможно заметить (отметим, что режим CBC имеет аналогичные ограничения, но невозможно заметить удаление данных в начале блока). Очевидным способом защиты или аутентификации

зашифрованного блока является цифровая подпись. Однако многие люди обычно не хотят подписывать данные по ряду причин, выходящих за рамки этого документа. Достаточно сказать, что многие считают такие свойства, как возможность отклонения (deniability), более важными, чем целостность.

OpenPGP решает задачу повышения уровня защищенности по сравнению с простым шифрованием, сохраняя возможность отклонения, с помощью системы MDC. MDC намеренно не является кодом аутентификации сообщения (Message Authentication Code или MAC) и название выбрано не случайно. Это аналог контрольной суммы.

Несмотря на то, что эта система достаточно скромна, она отлично зарекомендовала себя в реальном мире и эффективно защищает от нескольких атак, появившихся с момента её создания. Система прекрасно справляется со своими скромными задачами. Требования этой системы защиты к применяемой хэш-функции тоже достаточно скромны. Система не полагается на отсутствие конфликтов хэш-значений и ей достаточно необратимости хэш-функции. Если мошенник Франк захочет передать Алисе (цифровое) сообщение без подписи, где сказано: «Я всегда тайно любил вас! Боб», ему будет гораздо проще создать новое сообщение, нежели изменять перехваченное сообщение от Боба (отметим также, что если Боб захочет защищённо пообщаться с Алисой без аутентификации и идентификации, но с моделью угроз, учитывающей мошенников, у него возникнет проблема, выходящая за рамки простой криптографии).

Отметим также, что в отличие от почти всех других подсистем OpenPGP, в MDC не используется параметров. В ней жёстко задана хэш-функция SHA-1 и это не является случайностью. Это преднамеренный выбор, избавляющий от атак на понижение, а также перекрёстных атак и обеспечивающий простоту системы и высокую скорость. Атаками на понижение считаются атаки, заменяющие, например, SHA2-256 на SHA-1. Перекрёстная атака будет заменять SHA-1 другой 160-битовой хэш-функцией, например RIPEMD-160.

Обновления не требуется, поскольку система MDC заменена описанным в этом документе шифрованием AEAD.

5.13.2. Пакет с симметричным шифрованием и защитой целостности версии 2

Пакет с симметричным шифрованием и защитой целостности данных версии 2 состоит из:

- 1-октетного номера версии (2);
- 1-октетного идентификатора алгоритма шифрования;
- 1-октетного идентификатора алгоритма AEAD;
- 1-октетного размера блока (chunk);
- 32 октетов затравки (salt), используемой для вывода ключа сообщения; затравка должна создаваться защищённо (параграф 13.10);
- зашифрованных данных (вывод алгоритма шифрования с симметричным ключом в заданном режиме AEAD);
- итогового тега аутентификации для режима AEAD.

Расшифрованный сеансовый ключ и затравка применяются для получения M-битового ключа сообщения и N - 64 битов, служащих в качестве IV, где M - размер ключа симметричного алгоритма, а N - размер nonce алгоритма AEAD. M + N - 64 битов выводятся с использованием HKDF (см. [RFC5869]). M битов слева служат ключом симметричного алгоритма, а оставшиеся N - 64 битов - IV. HKDF применяется с алгоритмом хэширования SHA256 [RFC6234]. Сеансовый ключ применяется как IKM, а salt - как затравка. В качестве параметра info применяются Packet Type ID в формате OpenPGP (биты 7 и 6 установлены, а биты 5-0 содержат Packet Type ID), номер версии, идентификатор алгоритма шифрования, идентификатор алгоритма AEAD и октет размера блока.

Механизм KDF обеспечивает разделение ключей между алгоритмами шифрования и AEAD. Реализация может безопасно отвечать на сообщение даже при неизвестном сертификате получателя, повторно используя пакеты Encrypted Session Key с другой затравкой, которая даёт новый, уникальный ключ сообщения. В параграфе 13.8 описано, как приложения могут безопасно реализовать эту функцию.

Пакет SEIPD v2 содержит один или несколько блоков данных (chunk). Размер открытых данных каждого блока указывается отдельным октетом, как описано выше. Зашифрованные данные состоят из шифрованных блоков для каждого открытого блока, за которым следует тег аутентификации. Если размер последних открытых данных меньше размера блока, шифрованные данные могут быть короче, но за ними все равно следует полный тег аутентификации. Для каждого блока (chunk) конструкции AEAD в качестве дополнительных данных передаётся Packet Type ID в формате OpenPGP (биты 7 и 6 установлены, а биты 5-0 содержат Packet Type ID), номер версии, идентификатор алгоритма шифрования, идентификатор алгоритма AEAD и октет размера блока. Например, для первого блока размером 2^22 октетов, использующего EAX и AES-128, это будет последовательность октетов 0xD2, 0x02, 0x07, 0x01, 0x10.

После финального блока применяется алгоритм AEAD для создания финального тега аутентификации, шифрующего пустую строку. Этому экземпляру AEAD передаются указанные выше дополнительные данные, а также 8-октетное значение big-endian, указывающее общее число зашифрованных октетов открытых данных. Это позволяет обнаруживать отсечку шифрованных данных. Октет размера блока задаёт размер с использованием приведённой ниже формулы (язык C [C99]), где c - октет размера блока.

```
chunk_size = (uint32_t) 1 << (c + 6)
```

Реализации **должны** принимать октеты размера блока со значениями от 0 до 16 и **недопустимо** создавать данные с октетом размера больше 16 (4 Мбайта).

Значение nonce для AEAD состоит из двух частей. Пусть N - размер nonce, тогда N - 64 битов слева будут IV, полученным с помощью HKDF, а 64 бита справа - индексом блока в формате big-endian (отсчёт с 0).

5.13.3. Режим EAX

Применяемый в документе алгоритм AEAD EAX определён в [EAX] и может использовать только блочные шифры с размером блока 16 октетов. Значение nonce и тег аутентификации EAX имеют размер 16 октетов.

5.13.4. Режим OCB

Применяемый в документе алгоритм AEAD OCB определён в [RFC7253] и может использовать только блочные шифры с размером блока 16 октетов. Значение nonce имеет размер 15 октетов, тег аутентификации OCB - 16 октетов.

5.13.5. Режим GCM

Применяемый в документе алгоритм AEAD GCM определён в [SP800-38D] и может использовать только блочные шифры с размером блока 16 октетов. Значение nonce имеет размер 12 октетов, тег аутентификации GCM - 16 октетов.

5.14. Пакет заполнения (тип 21)

Пакеты Padding содержат случайные данные и могут служить для защиты от анализа трафика (параграф 13.11) сообщений SEIPD v2 (параграф 5.13.2) и ключей Transferable Public Key (параграф 10.1). Такие пакеты **должны** игнорироваться при получении. Содержимое пакетов **следует** делать случайным, чтобы обеспечиваемое ими сокрытие размера было более надёжным даже при использовании сжатия.

Реализациям, добавляющим заполнение в поток OpenPGP, **следует** помещать такие пакеты:

- в конце ключей Transferable Public Key версии 6, передаваемых по шифрованному каналу (параграф 10.1);
- в качестве последнего пакета Optionally Padded Message внутри пакетов SEIPD v2 (параграф 10.3.1).

Реализации **должны** поддерживать обработку пакетов Padding в любом месте потока OpenPGP, так как будущие версии этого документа могут задать иное размещение таких пакетов. Правила определения размера пакетов заполнения в целях защиты от анализа трафика выходят за рамки этого документа.

6. Преобразования Base64

Как отмечено в введении, базовым представлением объектов в OpenPGP является поток произвольных октетов и некоторые системы хотят защитить такие объекты от повреждений, связанных с трансляцией кодировок, преобразованием данных и т. п. В принципе, подходит любая система кодирования с возможностью печати, соответствующая требованиям для работы по незащищённым каналам, поскольку она не будет менять базовые битовые потоки структур данных OpenPGP. Спецификация OpenPGP задаёт одну из таких схем кодирования для обеспечения функциональной совместимости (см. параграф 6.2).

Кодированные данные состоят из двух частей - код base64 и необязательная контрольная сумма. Кодирование base64 описано в разделе 4 [RFC4648] и результаты представляются строками символов, размером не более 76. При декодировании base64 реализации OpenPGP **должны** игнорировать все пробельные символы.

6.1. Необязательная контрольная сумма

Необязательная контрольная сумма - это 24-битовое значение CRC (Cyclic Redundancy Check), представленное 4 символами base64, полученными с помощью того же преобразования MIME base64 с добавлением в начале знака равенства (=). Значение CRC рассчитывается с использованием генератора 0x864CFB и инициализации 0xB704CE. Накопление выполняется для данных до их преобразования в base64, а не для преобразованных данных. Простая реализация алгоритма представлена в параграфе 6.1.1.

При наличии контрольной суммы со знаком равенства в начале она **должна** указываться в следующей строке после данных в коде base64.

Реализациям **недопустимо** отвергать объекты OpenPGP по причине наличия, отсутствия, некорректного формата или несоответствия нижнего колонтитула CRC24 с рассчитанным значением CRC24. При формировании ASCII Armor колонтитул CRC24 **не следует** генерировать, если не нужна совместимость с реализациями, требующими его наличия. Колонтитул CRC24 **недопустимо** генерировать, если из контекста или объекта OpenPGP можно определить, что принимающая реализация воспринимает блоки base64 без колонтитула CRC24. Отметим, что:

- в последовательности пакетов зашифрованных сообщений с ASCII Armor, завершающиеся пакетом SEIPD v2, **недопустимо** включать колонтитул CRC24;
- в последовательности пакетов Signature с ASCII Armor, включающие только пакеты версии 6, **недопустимо** включать колонтитул CRC24;
- в последовательности пакетов Transferable Public Key с ASCII Armor и ключами версии 6 **недопустимо** включать колонтитул CRC24;
- в связки ключей с ASCII Armor и ключами только версии 6 **недопустимо** включать колонтитул CRC24.

В предшествующих черновых версиях документа говорилось, что колонтитул CRC24 является необязательным, но текст был неоднозначным. На практике лишь немногие реализации требуют наличия колонтитула CRC24. Расчёт CRC24 вносит существенные издержки, не обеспечивая значимой защиты целостности, поэтому создавать контрольные суммы в настоящее время не рекомендуется.

6.1.1. Реализация CRC24 на языке C

Ниже приведён код [C99].

```
#define CRC24_INIT 0xB704CEL
#define CRC24_GENERATOR 0x864CFBL

typedef unsigned long crc24;
crc24 crc_octets(unsigned char *octets, size_t len)
{
    crc24 crc = CRC24_INIT;
    int i;
    while (len--) {
```

```
            crc ^= (*octets++) << 16;
            for (i = 0; i < 8; i++) {
                crc <<= 1;
                if (crc & 0x1000000) {
                    crc &= 0XFFFFFF; /* Сброс бита 25 во избежание переполнения */
                    crc ^= CRC24_GENERATOR;
                }
            }
        }
        return crc & 0xFFFFFFL;
    }
```

6.2. Формирование ASCII Armor

При кодировании данных в ASCII Armor OpenPGP размещает специальные заголовки по обе стороны данных base64, чтобы их можно было восстановить позднее. Реализация OpenPGP **может** использовать ASCII Armor для защиты необработанных (raw) двоичных данных. OpenPGP информирует пользователя о типе данных в ASCII Armor с помощью заголовков. Для создания ASCII Armor используется конкатенация указанных ниже элементов:

- строка заголовка (Armor Header Line) в соответствии с типом данных;
- заголовки Armor;
- пустая строка или строка пробелов;
- данные с ASCII Armor;
- необязательная контрольная сумма Armor Checksum (устарела, см. параграф 6.1);
- трейлер Armor Tail в соответствии с типом данных.

6.2.1. Строка Armor Header

Armor Header Line содержит текст соответствующего заголовка окружённого последовательностями из 5 символов тире (-, 0x2D) в начале и в конце. Текст строки заголовка определяется типом данных, кодируемых в Armor и их последующим кодированием. Список текстов строк заголовка приведён в таблице.

Таблица 16. Реестр строк заголовка OpenPGP Armor.

Armor Header Line	***Применение***
BEGIN PGP MESSAGE	Для подписанных, шифрованных или сжатых файлов.
BEGIN PGP PUBLIC KEY BLOCK	Для защиты открытых ключей.
BEGIN PGP PRIVATE KEY BLOCK	Для защиты секретных ключей.
BEGIN PGP SIGNATURE	Для отдельных подписей, а также подписей OpenPGP/MIME и открытых.

Отметим, что Armor Header Line является отдельной строкой, поэтому заголовок **должен** начинаться с новой строки и **недопустимо** включать в строку какой либо текст, за исключением пробельных символов.

6.2.2. Заголовки Armor

Armor Header - это пары строк, которые могут давать пользователю или принимающей реализации OpenPGP некоторые сведения о способе декодирования и использования сообщения. Armor Header - это часть защиты, а не сообщения, поэтому они не учитываются в подписях сообщений.

Формат Armor Header - это ключ и значение, разделённые двоеточием (: 0x3A) и одним пробелом (0x20). Реализация OpenPGP может считать некорректно сформированные Armor Header повреждением ASCII Armor и ей **следует** принять меры по восстановлению. Неизвестные ключи следует просто игнорировать, а реализации OpenPGP с таком случае **следует** продолжать обработку сообщения.

Следует отметить, что некоторые методы доставки чувствительны к размеру строк. Например, протокол SMTP, доставляющий сообщения электронной почты, имеет ограничение размера строк в 998 (параграф 2.1.1 в [RFC5322]).

В то время, как размер строк base64 ограничен 76 символами (раздел 6), для размера Armor Header ограничений не задано. Следует позаботиться о том, чтобы размер Armor Header не нарушал требования транспорта. Одним из способов достижения этого является повтор Armor Header Key в нескольких строках с единственным значением в строке, чтобы ни одна строка не была слишком длинной.

Определённые в настоящее время Armor Header Key указаны в таблице.

Таблица 17. Реестр ключей заголовка OpenPGP Armor.

Ключ	***Сводка***	***Документ***
Version	Сведения о реализации	параграф 6.2.2.1
Comment	Произвольный текст	параграф 6.2.2.2
Hash	Алгоритмы хэширования, применяемые в некоторых сообщениях v4 с открытой подписью	параграф 6.2.2.3
Charset	Кодировка (набор символов)	параграф 6.2.2.4

6.2.2.1. Заголовок Version

Ключ заголовка Version описывает реализацию OpenPGP и версию, использованную для кодирования сообщения. Для минимизации метаданных реализациям **не следует** использовать этот ключ и соответствующее ему значение за исключением случаев отладки с явного согласия пользователя.

6.2.2.2. Заголовок Comment

Заголовок Comment содержит заданный пользователем комментарий. В OpenPGP текст комментария должен иметь кодировку UTF-8 и комментарий может быть любой строкой UTF-8. Однако смысл защиты (armoring) заключается в том, что бит 7 в символах сброшен (0), поэтому комментарии, содержащие символы UTF-8 за пределами диапазона кодов ASCII, могут не пройти через транспорт.

6.2.2.3. Заголовок Hash

Заголовок Hash признан устаревшим, но некоторые старые реализации ожидают его в сообщениях, использующих Cleartext Signature Framework (раздел 7). При наличии заголовка он содержит список (через запятую) алгоритмов

хеширования, используемых в подписях сообщения, с именами дайджестов, указанными в таблице 23. Эти заголовки **не следует** применять, если не выполняются указанные ниже условия:

- сообщение содержит открытую (cleartext) подпись версии 4, выполненную с использованием дайджеста SHA2 (SHA224, SHA256, SHA384, SHA512);
- сообщение с открытой подписью может быть проверено унаследованной реализацией OpenPGP, которая требует наличия этого заголовка.

Проверяющее приложение **должно** отвергать любую подпись в сообщении, содержащую несоответствующий заголовок Hash (заголовок, включающий что-либо сверх списка алгоритмов хэширования через запятую). При наличии соответствующего заголовка Hash проверяющее приложение **должно** игнорировать его содержимое, обращаясь к алгоритму хэширования, указанному в Embedded Signature.

6.2.2.4. Заголовок Charset

Заголовок Charset содержит описание набора символов (кодировки), использованного в открытом тексте (см. [RFC2978]). Отметим, что OpenPGP задаёт для текста кодировку UTF-8. Реализация будет лучше работать, переводя текст в кодировку UTF-8 (и обратно). Однако во многих случаях это легче сказать, чем сделать. Кроме того, некоторым сообществам пользователей не нужна кодировка UTF-8, поскольку их устраивает иной набор символов, например, ISO Latin-5 или Japanese. В таких случаях реализация **может** переопределить принятую по умолчанию кодировку UTF-8 с помощью этого заголовка. Реализация **может** применять этот ключ заголовка и любые нужные трансляции символов, а также **может** игнорировать ключ и считать, что текст использует кодировку UTF-8.

6.2.3. Защитный трейлер

Строка трейлера Armor Tail Line собирается так же, как Armor Header Line, но с заменой BEGIN на END.

7. Подпись в форме открытого текста

Желательна возможность подписывать поток октетов текста без ASCII-защиты самого потока, чтобы подписанный текст можно было прочесть любым инструментом, способным разбирать текст. Для привязки подписи к такому открытому тексту применяется схема открытой подписи (Cleartext Signature Framework), имеющая такой же базовый формат и ограничения, как описанная в параграфе 6.2 защита ASCII. Отметим, что этот механизм не предполагается обратимым, а в [RFC3156] определён иной способ подписывания сообщений с открытым тестом для сред, поддерживающих MIME.

7.1. Структура открытого сообщения с подписью

Подписанное открытое сообщение OpenPGP состоит из:

- заголовка открытого текста -----BEGIN PGP SIGNED MESSAGE----- в виде одной строки;
- одного или нескольких унаследованных заголовков Hash Armor, которые **могут** включаться некоторыми реализациями и **должны** игнорироваться при корректном формировании (см. параграф 6.2.2.3);
- пустой строки (не включается в дайджест сообщения);
- открытого текста с экранированием дефисами (dash-escaped);
- завершения строки, отделяющего открытый текст от следующей за ним защищённой подписи (не включается в дайджест сообщения);
- подписи с ASCII-защитой, включающей заголовок -----BEGIN PGP SIGNATURE----- и трейлер защиты.

Как и подпись любого другого текста (параграф 5.2.1.2), подпись открытого текста рассчитывается для текста с завершением строк в форме <CR><LF>. Как описано выше, строки перед -----BEGIN PGP SIGNATURE----- (Armor Header Line) не считается частью подписываемого текста. Пробельные символы - пробелы (0x20) и табуляторы (0x09) - в конце любой строки удаляются перед созданием или проверкой открытого сообщения с подписью.

Между строкой -----BEGIN PGP SIGNED MESSAGE----- и первой пустой строкой единственным разрешённым заголовком Armor является корректно сформированный заголовок Hash (параграф 6.2.2.3). Для снижения риска путаницы в отношении того, что подписано, проверяющая реализация **должна** отказываться от проверки подписи в открытом сообщении, если в этом месте присутствует какой-либо иной заголовок Armor.

7.2. Текст с экранированием дефисами

В открытом содержимом сообщения должно использоваться экранирование дефисом (dash-escape). Открытый текст с экранированием дефисом - это оригинальный открытый текст, в котором каждая строка, начинающаяся с дефиса (-) (HYPHEN-MINUS, U+002D), имеет префикс в виде дефиса и пробела (SPACE, U+0020) «- ». Это предотвращает распознавание синтаксическим анализатором заголовков Armor как открытого текста. Реализация **может** использовать экранирование в любой строке, **следует** экранировать строки, начинающиеся с «From » и **должны** экранироваться все строки, начинающиеся с дефиса. Дайджест сообщения рассчитывается только для самого открытого текста без включения символов экранирования.

При снятии экранирования реализация **должна** вырезать дефис в начале строки, а при наличии в начале строки дефиса, за которым следует любой символ, кроме пробела, **следует** выдавать предупреждение.

7.3. Проблемы схемы подписи открытого текста

Поскольку создание подписанного открытого текста включает удаление пробелов в конце каждой строки, схема Cleartext Signature не может безопасно обрабатывать текстовые потоки с семантически важными пробелами. Например, формат Unified Diff [UNIFIED-DIFF] содержит семантически важные пробелы - пустая строка контекста будет состоять из одного символа пробела « » (SPACE, U+0020) и любая строка с добавленным или удаленным пробелом в конце будет представлять такое изменение с семантически важными пробелами.

Кроме того, очень большое сообщение Cleartext Signature вряд ли будет работать хорошо. В частности, читающему сообщение человеку будет трудно понять, какая часть сообщения охвачена подписью, поскольку он не сможет понять все сообщение сразу, особенно при наличии в нем строки Armor Header. Очень большое сообщение Cleartext Signature не может быть обработано за один проход, поскольку затравка (salt) подписи и алгоритмы дайджеста обнаруживаются лишь в конце.

Реализации, осознающей, что она работает с текстовым сообщением, обладающим любой из указанных выше характеристик, **не следует** применять схему Cleartext Signature. Безопасными альтернативами для семантически важной подписи OpenPGP файлов такого формата являются:

- сообщение, подписанное в соответствии с параграфом 10.3;
- отдельная подпись, как описано в параграфе 10.4.

В любом из этих вариантов можно использовать ASCII Armor (параграф 6.2) , если требуется передача по каналу, поддерживающему только текст (7-битовые символы).

Когда сообщение Cleartext Signature представляется пользователю как есть, злоумышленник может включить в заголовок Hash дополнительный текст, который пользователь воспримет как часть подписанного текста. Ограничения при проверке подписи, описанные в параграфах 6.2.2.3 и 7.1, помогут снизить риск появления произвольного или вводящего в заблуждение текста в заголовках Armor.

8. Регулярные выражения

Regular Expression - регулярное выражение

Ветви (возможно, 0), разделённые символом |. Регулярному выражению соответствует все, что соответствует любой из его ветвей.

Branch - ветвь

Конкатенация (возможно, пустая) частей. Ветви соответствуют совпадения для каждой из частей по порядку.

Piece - часть

Атом, за которым может следовать *, +, или ?. Атому со звёздочкой (*) соответствует последовательность (возможно пустая) совпадений с ним. Атому со знаком + соответствует последовательность из одного или нескольких совпадений с ним. Атому со знаком вопроса (?) соответствует совпадение с ним или пустая строка.

Atom - атом (неделимый блок)

Регулярное выражение в круглых скобках (соответствует совпадению с Regular Expression), диапазон, точка (.) (соответствует одному символу Unicode), ^ (соответствует null-строке в начале входной строки), $ (соответствует null-строке в конце входной строки), \ с одним символом Unicode (соответствует этому символу) или один символ Unicode, не имеющий другого значения (соответствует этому символу).

Range - диапазон

Последовательность символов в квадратных скобках []. Обычно диапазон соответствует любому одиночному символу из последовательности. Если последовательность начинается с ^, ей соответствует любой одиночный символ Unicode, не включенный в остальную часть последовательности. Если два символа последовательности разделены дефисом (-), это является сокращением для всех символов Unicode между указанными символами в порядке их кодов (например, [0-9] соответствует любая десятичная цифра). Для включения в последовательность символа] его нужно сделать первым (после возможного символа ^), для включения дефиса (-) его нужно сделать первым или последним.

9. Константы

Приведённые в таблицах списки не являются исчерпывающими и реализации **могут** применять не указанные здесь алгоритмы, а также алгоритмы из диапазона Private or Experimental Use. Дополнительные сведения об алгоритмах приведены в разделе 12.

9.1. Алгоритмы с открытым ключом

Таблица 18. Реестр алгоритмов OpenPGP с открытым ключом

ID	*Алгоритм*	*Формат*			
		Открытый ключ	*Секретный ключ*	*Подпись*	*PKESK*
0	Резерв				
1	RSA (шифрование или подписи) [FIPS186]	MPI(n), MPI(e) [параграф 5.5.5.1]	MPI(d), MPI(p), MPI(q), MPI(u)	MPI(m^d mod n) [параграф 5.2.3.1]	MPI(m^e mod n) [параграф 5.1.3]
2	RSA (только шифрование) [FIPS186]	MPI(n), MPI(e)[параграф 5.5.5.1]	MPI(d), MPI(p), MPI(q), MPI(u)	-	MPI(m^e mod n) [параграф 5.1.3]
3	RSA (только подписи) [FIPS186]	MPI(n), MPI(e) [параграф 5.5.5.1]	MPI(d), MPI(p), MPI(q), MPI(u)	MPI(m^d mod n) [параграф 5.2.3.1]	-
16	Elgamal (только шифрование) [ELGAMAL]	MPI(p), MPI(g), MPI(y) [параграф 5.5.5.3]	MPI(x)	-	MPI(g^k mod p), MPI(m * y^k mod p) [параграф 5.1.4]
17	DSA (Digital Signature Algorithm) [FIPS186]	MPI(p), MPI(q), MPI(g), MPI(y) [параграф 5.5.5.2]	MPI(x)	MPI(r), MPI(s) [параграф 5.2.3.2]	-
18	Алгоритм с открытым ключом ECDH	OID, MPI(точка в зависимом от кривой формате), KDFParams [параграфы 9.2.1 и 5.5.5.6]	MPI(значение в зависимом от кривой формате) [параграф 9.2.1]	-	MPI(точка в зависящем от кривой формате), октет размера, кодированный ключ [параграфы 9.2.1, 5.1.5 и 11.5]
19	Алгоритм с открытым ключом [FIPS186]	OID, MPI(точка в формате SEC1) [параграф 5.5.5.4]	MPI(value)	MPI(r), MPI(s) [параграф 5.2.3.2]	N/A
20	Резерв (ранее Elgamal для шифрования или подписи)				

21	Резерв для Diffie-Hellman (X9.42, как для IETF-S/MIME)				
22	EdDSALegacy (устарел)	OID, MPI(точка в естественном формате с префиксом) [параграфы 11.2.2 и 5.5.5.5]	MPI(значение в зависимом от кривой формате) [параграф 9.2.1]	MPI, MPI [параграфы 9.2.1 и 5.2.3.3]	N/A
23	Резерв (AEDH)				
24	Резерв (AEDSA)				
25	X25519	32 октета [параграф 5.5.5.7]	32 октета	-	32 октета, октет размера, кодированный ключ [параграф 5.1.6]
26	X448	56 октетов [параграф 5.5.5.8]	56 октетов	-	56 октетов, октет размера, кодированный ключ [параграф 5.1.7]
27	Ed25519	32 октета [параграф 5.5.5.9]	32 октета	64 октета [параграф 5.2.3.4]	
28	Ed448	57 октетов [параграф 5.5.5.10]	57 октетов	114 октетов [параграф 5.2.3.5]	
100 - 110	Приватное использование и эксперименты				

Реализации **должны** поддерживать Ed25519 (27) для подписей и X25519 (25) для шифрования, **следует** также поддерживать Ed448 (28) и X448 (26).

Ключи RSA (1) устарели и их **не следует** генерировать, но можно интерпретировать. RSA Encrypt-Only (2) и RSA Sign-Only (3) устарели и генерировать их **недопустимо** (параграф 12.4). Ключи Elgamal (16) устарели и генерировать их **недопустимо** (параграф 12.6). Ключи DSA (17) устарели и генерировать их **недопустимо** (параграф 12.5). Замечания для Elgamal Encrypt или Sign (20) и X9.42 (21) приведены в параграфе 12.8. Реализации **могут** поддерживать любые другие алгоритмы.

Отметим, что реализации, соответствующие прежней спецификации [RFC4880], включают лишь DSA (17) и Elgamal (16) в качестве алгоритмов, которые **должны** поддерживаться.

Совместимая спецификация ECDSA приведена в [RFC6090] (как подписи KT-I) и [SEC1], метод ECDH задан в параграфе 11.5 этого документа.

9.2. Кривые ECC для OpenPGP

Параметр Curve OID - это массив октетов, определяющих именованную кривую. В таблице 19 приведены последовательности октетов для всех именованных кривых, рассмотренных в этом документе. Указаны также применяемые ими алгоритмы с открытым ключом и размеры ожидаемых элементов в октетах.

Таблица 19. Реестр OpenPGP ECC Curve OID и применений.

Идентификатор объекта ASN.1	***Размер OID***	***Октеты Curve OID***	***Имя кривой***	***Применение***	***Размер поля (fsize)***
1.2.840.10045.3.1.7	8	2A 86 48 CE 3D 03 01 07	NIST P-256	ECDSA, ECDH	32
1.3.132.0.34	5	2B 81 04 00 22	NIST P-384	ECDSA, ECDH	48
1.3.132.0.35	5	2B 81 04 00 23	NIST P-521	ECDSA, ECDH	66
1.3.36.3.3.2.8.1.1.7	9	2B 24 03 03 02 08 01 01 07	brainpoolP256r1	ECDSA, ECDH	32
1.3.36.3.3.2.8.1.1.11	9	2B 24 03 03 02 08 01 01 0B	brainpoolP384r1	ECDSA, ECDH	48
1.3.36.3.3.2.8.1.1.13	9	2B 24 03 03 02 08 01 01 0D	brainpoolP512r1	ECDSA, ECDH	64
1.3.6.1.4.1.11591.15.1	9	2B 06 01 04 01 DA 47 0F 01	Ed25519Legacy	EdDSALegacy	32
1.3.6.1.4.1.3029.1.5.1	10	2B 06 01 04 01 97 55 01 05 01	Curve25519Legacy	ECDH	32

Столбец «Размер поля (fsize)» указывает размер поля группы в октетах, округлённый вверх так, что координаты x или y для точки кривой или естественные представления точек кривой могут быть указаны таким числом октетов. Заданные здесь кривые и скаляры, такие как порядок базовой точки и секретный ключ, можно представить, используя fsize октетов. Однако отметим, что имеются кривые, не включённые в спецификацию, где для представления скаляров требуется дополнительный октет.

Последовательность октетов в третьем столбце является результатом применения правил отличительного кодирования (Distinguished Encoding Rules или DER) к ASN.1 OID с последующей отсечкой, удаляющей два поля кодированного идентификатора объекта. Первым исключаемым полем является октет, представляющий тег OID, а вторым - размер тела OID. Например, полное декодирование ASN.1 DER для OID кривой NIST P-256 - это последовательность 06 08 2A 86 48 CE 3D 03 01 07, из которой создаётся запись в первой строке таблицы путём исключения двух первых октетов. Действительным представлением OID кривой является лишь усечённая последовательность.

Устаревшие OID для Ed25519Legacy и Curve25519Legacy используются только в ключах и подписях версии 4. Реализации **могут** поддерживать эти варианты для совместимости с имеющимся ключевым материалом и подписями версии 4. Реализациям **недопустимо** воспринимать или генерировать ключевой материал версии 6 с использованием устаревших OID.

9.2.1. Зависящие от кривых форматы передачи

Некоторые алгоритмы с открытым ключом на основе эллиптической кривой используют разные соглашения для определённых полей в зависимости от применяемой кривой. Поля всегда имеют формат MPI, но структура зависит от кривой. Различия показаны в таблице 20.

Таблица 20. Реестр зависящих от кривой ECC форматов передачи.

Кривая	Формат точки ECDH	ECDH Secret Key MPI	EdDSA Secret Key MPI	EdDSA Signature MPI 1	EdDSA Signature MPI 2
NIST P-256	SEC1	integer	-	-	-
NIST P-384	SEC1	integer	-	-	-
NIST P-521	SEC1	integer	-	-	-
brainpoolP256r1	SEC1	integer	-	-	-
brainpoolP384r1	SEC1	integer	-	-	-
brainpoolP512r1	SEC1	integer	-	-	-
Ed25519Legacy	-	-	32 октета секрета	32 октета R	32 октета S
Curve25519Legacy	Естественный с префиксом	integer (параграф 5.5.5.6.1.1)	-	-	-

Естественные формы строк октетов для значений Ed25519Legacy представлены в [RFC8032], для секретных скаляров и точек Curve25519Legacy - в [RFC7748].

9.3. Симметричные алгоритмы

Таблица 21. Реестр симметричных алгоритмов OpenPGP.

ID	Алгоритм
0	Открытый текст или незашифрованные данные
1	IDEA [IDEA]
2	TripleDES (или DES-EDE) [SP800-67] с выводом 168-битового ключа из 192
3	CAST5 со 128-битовым ключом [RFC2144]
4	Blowfish со 128-битовым ключом, 16 раундов [BLOWFISH]
5	Резерв
6	Резерв
7	AES со 128-битовым ключом [AES]
8	AES со 192-битовым ключом
9	AES с 256-битовым ключом
10	Twofish с 256-битовым ключом [TWOFISH]
11	Camellia со 128-битовым ключом [RFC3713]
12	Camellia со 192-битовым ключом
13	Camellia с 256-битовым ключом
100-110	Приватное использование и эксперименты
253-255	Резерв для предотвращения конфликтов с Secret Key Encryption (таблица 2 и параграф 5.5.3)

Реализации **должны** поддерживать AES-128, **следует** поддерживать AES-256. **Недопустимо** шифровать данные с использованием IDEA, TripleDES или CAST5, но **можно** расшифровывать такие данные для чтений старых или новых сообщений от реализаций, предшествующих поддержке [RFC2440]. Реализациям с расшифровкой IDEA, TripleDES или CAST5 **следует** выдавать предупреждение об устаревшем симметричном алгоритме, указывающее, что конфиденциальность полученного сообщения сомнительна. Реализации **могут** поддерживать любые другие алгоритмы.

9.4. Алгоритмы сжатия

Таблица 22. Реестр алгоритмов сжатия OpenPGP.

ID	Алгоритм
0	Без сжатия
1	ZIP [RFC1951]
2	ZLIB [RFC1950]
3	BZip2 [BZ2]
100-110	Приватное использование и эксперименты

Реализации **должны** поддерживать данные без сжатия и **следует** поддерживать ZLIB. Для совместимости **следует** поддерживать декомпрессию ZIP. Реализации **могут** поддерживать любые другие алгоритмы.

9.5. Алгоритмы хэширования

Таблица 23. Реестр алгоритмов хэширования OpenPGP.

ID	Алгоритм	Имя	Размер затравки подписи V6
0	Резерв		
1	MD5 [RFC1321]	MD5	-
2	SHA-1 [FIPS180]	SHA1	-
3	RIPEMD-160 [RIPEMD-160]	RIPEMD160	-
4	Резерв		
5	Резерв		
6	Резерв		
7	Резерв		
8	SHA2-256 [FIPS180]	SHA256	16
9	SHA2-384 [FIPS180]	SHA384	24
10	SHA2-512 [FIPS180]	SHA512	32
11	SHA2-224 [FIPS180]	SHA224	16
12	SHA3-256 [FIPS202]	SHA3-256	16
13	Резерв		
14	SHA3-512 [FIPS202]	SHA3-512	32
100-110	Приватное использование и эксперименты		

Таблица 24. Реестр идентификаторов алгоритмов хэширования OpenPGP для заполнения EMSA-PKCS1-v1_5 с подписями RSA.

Алгоритм	***OID***	***Полный хэш-префикс***
MD5	1.2.840.113549.2.5	0x30, 0x20, 0x30, 0x0C, 0x06, 0x08, 0x2A, 0x86, 0x48, 0x86, 0xF7, 0x0D, 0x02, 0x05, 0x05, 0x00, 0x04, 0x10
SHA-1	1.3.14.3.2.26	0x30, 0x21, 0x30, 0x09, 0x06, 0x05, 0x2B, 0x0E, 0x03, 0x02, 0x1A, 0x05, 0x00, 0x04, 0x14
RIPEMD-160	1.3.36.3.2.1	0x30, 0x21, 0x30, 0x09, 0x06, 0x05, 0x2B, 0x24, 0x03, 0x02, 0x01, 0x05, 0x00, 0x04, 0x14
SHA2-256	2.16.840.1.101.3.4.2.1	0x30, 0x31, 0x30, 0x0D, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x01, 0x05, 0x00, 0x04, 0x20
SHA2-384	2.16.840.1.101.3.4.2.2	0x30, 0x41, 0x30, 0x0D, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x02, 0x05, 0x00, 0x04, 0x30
SHA2-512	2.16.840.1.101.3.4.2.3	0x30, 0x51, 0x30, 0x0D, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x03, 0x05, 0x00, 0x04, 0x40
SHA2-224	2.16.840.1.101.3.4.2.4	0x30, 0x2D, 0x30, 0x0D, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x04, 0x05, 0x00, 0x04, 0x1C
SHA3-256	2.16.840.1.101.3.4.2.8	0x30, 0x31, 0x30, 0x0D, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x08, 0x05, 0x00, 0x04, 0x20
SHA3-512	2.16.840.1.101.3.4.2.10	0x30, 0x51, 0x30, 0x0D, 0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03, 0x04, 0x02, 0x0a, 0x05, 0x00, 0x04, 0x40

Реализации **должны** поддерживать SHA2-256, **следует** поддерживать SHA2-384 и SHA2-512, а также **можно** поддерживать другие алгоритмы. Реализациям **не следует** создавать сообщения, требующие использовать SHA-1, за исключением расчёта оттисков ключей версии 4 для целей MDC в пакетах данных с симметричным шифрованием и защитой целостности версии 1. Реализациям **недопустимо** создавать подписи с MD5, SHA-1, RIPEMD-160. **Недопустимо** использовать MD5, SHA-1, RIPEMD-160 в качестве хэш-функции для ECDH KDF и для S2K KDF. Реализациям **недопустимо** расшифровывать секреты, использующие MD5, SHA-1, RIPEMD-160 в качестве хэш-функции для S2K KDF в пакетах версии 6 (или выше). **Недопустимо** подтверждать (validate) недавние подписи, зависящие от MD5, SHA-1, RIPEMD-160. **Не следует** подтверждать какие-либо старые подписи, зависящие от MD5, SHA-1, RIPEMD-160, за исключением случаев, когда дата подписи раньше обнаружения уязвимостей использованного алгоритма и реализация уверена, что сообщение все это время надёжно хранилось у пользователя.

9.6. Алгоритмы AEAD

Таблица 25. Реестр алгоритмов OpenPGP AEAD.

ID	***Имя***	***Размер Nonce***	***Размер тега аутентификации***
0	Резерв		
1	EAX [EAX]	16 октетов	16 октетов
2	OCB [RFC7253]	15 октетов	16 октетов
3	GCM [SP800-38D]	12 октетов	16 октетов
100-110	Приватное использование и эксперименты		

Реализации **должны** поддерживать OCB и **могут** поддерживать EAX, GCM и другие алгоритмы.

10. Организация последовательности пакетов

Пакеты OpenPGP собираются в последовательности для образования сообщений и передачи ключей. Не все последовательности пакетов являются значимыми и корректными и в этом разделе приводятся правила размещения пакетов в последовательности. Существует три различных последовательности пакетов:

- переносимые открытые (параграф 10.1) и секретные (параграф 10.2) ключи;
- сообщения OpenPGP (параграф 10.3);
- отдельные (отсоединённые) подписи (параграф 10.4).

Для каждой последовательности имеется явная грамматика типов пакетов (таблица 3) и их размещения. Наличие неизвестного критического пакета или известного, но неожиданного, является критической ошибкой, делающей недействительной всю последовательность (параграф 4.3). С другой стороны, неизвестные некритические пакеты могут появляться в любом месте последовательности. Это обеспечивает структурированный способ введения в OpenPGP новых пакетов с гарантией строгой обработки некоторых пакетов.

Реализация может распознавать пакет, но не поддерживать его. Критичность пакетов (Packet Criticality) позволяет сообщить потребителю предпочтительный вариант при получении нового, неизвестного пакета - считать это ошибкой или просто игнорировать пакет. Отметим, что в предыдущих версиях документа не было понятия критичности пакета и чётких рекомендаций по действиям при обнаружении неизвестных пакетов. Поэтому реализации прежних версий могут отвергать некритические пакеты или воспринимать неизвестные критичные пакеты.

При создании последовательности пакетов OpenPGP в соответствии с одним из трёх вариантов грамматики реализации **недопустимо** внедрять критические пакеты не соответствующего в грамматике типа.

Если при обработке последовательности пакетов OpenPGP реализация встречает критический пакет не соответствующего грамматике типа, она **должна** отклонить последовательность с ошибкой.

10.1. Переносимые открытые ключи

В OpenPGP могут применяться переносимые открытые ключи и в этом параграфе описывается структура открытых ключей при их передаче для обеспечения функциональной совместимости. Переносимые открытые ключи OpenPGP (Transferable Public Key) называют также сертификатами OpenPGP, чтобы отличать их как от составляющих пакетов Public Key (параграфы 5.5.1.1 и 5.5.1.2), так и от базового криптографического ключевого материала.

10.1.1. Структура сертификата OpenPGP версии 6

Формат сертификатов OpenPGP версии 6 показан ниже. В квадратные скобки заключены необязательные элементы, а многоточие указывает на повторение.

```
Primary Key
   [Revocation Signature...]
    Direct Key Signature…
   [User ID or User Attribute
           [Certification Revocation Signature...]
           [Certification Signature...]]…
   [Subkey [Subkey Revocation Signature...]
           Subkey Binding Signature...]…
   [Padding]
```

В дополнение к этому в любом месте последовательности может присутствовать пакет Marker (параграф 5.8).

Отметим, что в ключах версии 6 используется самоподпись Direct Key для сохранения предпочтений алгоритма.

Каждый субключ первичного ключа версии 6 **должен** иметь версию 6. Каждый субключ **должен** иметь хотя бы одну подпись Subkey Binding, а каждая такая подпись **должна** быть самоподписью (т. е. создана с первичным ключом версии 6). Как и все прочие подписи, самоподпись, созданная с ключом версии 6, **должна** быть подписью версии 6.

10.1.2. Сертификат отзыва OpenPGP версии 6

Отзыв первичного открытого ключа версии 6 иногда распространяется лишь с подписью Revocation Signature

```
Primary Key
    Revocation Signature
```

В этом случае подпись Direct Key не требуется, поскольку сам первичный ключ помечается как неиспользуемый.

10.1.3. Структура сертификата OpenPGP версии 4

Ниже показан формат ключа OpenPGP версии 4.

```
Primary Key
   [Revocation Signature]
   [Direct Key Signature...]
   [User ID или User Attribute [Signature...]]…
   [Subkey [Subkey Revocation Signature...]
           Subkey Binding Signature...]…
```

В дополнение к этому в любом месте последовательности может присутствовать пакет Marker (параграф 5.8).

После субключа всегда имеется хотя бы одна подпись Subkey Binding, выдаваемая для связывания двух ключей. Эти подписи привязки могут иметь формат версии 3 или 4, но **следует** использовать версию 4. Субключи, которые могут выпускать подписи, **должны** иметь подпись привязки версии 4 из-за **требования** иметь встроенную подпись Primary Key Binding.

Каждый субключ для первичного ключа версии 4 **должен** иметь версию 4.

При отзыве первичного Public Key версии 4 подпись отзыва (Revocation Signature) иногда распространяется сама по себе без пакета первичного ключа, к которому она относится. Это называется сертификатом отзыва (revocation certificate). Сертификат отзыва версии 6 **должен** включать пакет первичного ключа, как указано в параграфе 10.1.2.

10.1.4. Структура ключа OpenPGP версии 3

Ниже показан формат ключей OpenPGP версии 3.

```
RSA Public Key
   [Revocation Signature]
    User ID [Signature...]
   [User ID [Signature...]]...
```

В дополнение к этому в любом месте последовательности может присутствовать пакет Marker (параграф 5.8).

Каждая подпись сертифицирует открытый ключ RSA и предыдущий идентификатор пользователя (User ID). Открытый ключ RSA может иметь несколько User ID и каждый из таких идентификаторов может иметь не одну подпись. Ключи версии 3 устарели и реализациям **недопустимо** генерировать новые ключи версии 3, но **можно** продолжать использование имеющихся.

Ключам версии **недопустимо** иметь субключи.

10.1.5. Общие требования

Пакет Public Key указывается первым. У первичного ключа **должен** быть алгоритм, способный создавать подписи (т. е. не только шифровать), потому что от него требуется возможность заверять самого себя (см. параграф 5.2.3.10). Субключи могут иметь любой тип. Например, может быть один ключ RSA - первичный ключ Ed25519 с субключом шифрования RSA, первичный ключ Ed25519 с субключом X25519 и т. п.

Каждый из следующих пакетов User ID отождествляет владельца данного открытого ключа. Наличие нескольких пакетов User ID соответствует множеству способов идентификации одного и того же уникального пользователя, например, пользователь может иметь не один адрес электронной почты и создать User ID для каждого из них. Переносимому открытому ключу (Transferable Public Key) **следует** содержать хотя бы один пакет User ID, если это не запрещают требования хранилища.

Сразу после каждого пакета User ID могут следовать пакеты Signature, каждый из которых рассчитывается для предшествующего ему пакета User ID с исходным пакетом Public Key. Подпись служит для сертификации соответствующего открытого ключа и User ID. Фактически, подписывающий подтверждает свою уверенность в том, что этот открытый ключ относится к пользователю, указанному данным User ID.

В одном разделе с пакетами User ID могут присутствовать пакеты User Attribute, за каждым из которых могут следовать пакеты Signature, рассчитанные для непосредственно предшествующего пакета User Attribute с исходным пакетом Public Key. Пакеты User Attribute и User ID можно свободно перемешивать в разделе при условии, что размещённые вслед за пакетами подписи относятся к соответствующим пакетам User Attribute или User ID.

Вслед за пакетами User ID и User Attribute и связанными с ними подписями могут размещаться пакеты Subkey, каждый из которых имеет свои подписи. Обычно субключ предоставляются в случаях, когда открытый ключ верхнего уровня предназначен лишь для сертификации. Однако любые ключи версии 4 или 6 могут иметь субключи, которые могут быть ключами шифрования, подписи, аутентификации и т. п. Рекомендуется применять свои ключи для каждой операции (например, только для подписи, только для шифрования, только для аутентификации и т. д.).

За каждым пакетом Subkey **должен** следовать один пакет Signature, которому следует быть подписью привязки субключа (Subkey Binding), созданной с ключом верхнего уровня. Для субключей, которые могут выдавать подписи, в Subkey Binding **должен** содержаться субпакет Embedded Signature с подписью Primary Key Binding (Type ID 0x19), созданной с субключом ключа верхнего уровня.

За каждым пакетом Subkey и Key может следовать пакет Revocation Signature для индикации отзыва ключа. Пакеты Revocation Signature воспринимаются лишь при выдаче подписи самим ключом, или другим ключом, уполномоченным на отзывы с помощью субпакетов Revocation Key в самоподписи с ключом верхнего уровня.

Необязательные пакеты Padding в конце обеспечивают механизм защиты от анализа трафика (параграф 13.11). Для максимальной совместимости при использовании пакетов Public Key версии 4 пакет Padding **не следует** включать, пока получатель не указал, что он способен игнорировать такой пакет. Реализации, способные принимать Transferable Public Key с первичным ключом Public Key версии 6, **должны** быть способны воспринимать (и игнорировать) необязательный пакет Padding.

Последовательности пакетов Transferable Public Key можно объединять (конкатенация) для обеспечения возможности передачи нескольких открытых ключей в одной операции (параграф 3.6).

10.2. Переносимые секретные ключи

Пользователи OpenPGP могут передавать секретные ключи. Формат Transferable Secret Key отличается от Transferable Public Key лишь тем, что может включать пакеты Secret Key и Secret Subkey в дополнение к Public Key и Public Subkey. Если в последовательность пакетов включён один пакет Secret Key или Secret Subkey, это будет Transferable Secret Key, который следует обрабатывать и помечать как таковой (параграф 6.2.1). Реализации **следует** включать самоподписи для любых последующих User ID и субключей, поскольку это позволяет автоматически извлекать полный открытый ключ из Transferable Secret Key. Реализация **может** отказаться от самоподписей, особенно в случаях, когда Transferable Public Key сопровождается Transferable Secret Key.

10.3. Сообщения OpenPGP

Сообщение OpenPGP - это пакет или последовательность пакетов, соответствующих приведённым ниже правилам (запятая указывает последовательность, а вертикальная черта | разделяет варианты).

OpenPGP Message
: Encrypted Message | Signed Message | Compressed Message | Literal Message.

Compressed Message
: Compressed Data Packet.

Literal Message
: Literal Data Packet.

ESK
: Public Key Encrypted Session Key Packet | Symmetric Key Encrypted Session Key Packet.

ESK Sequence
: ESK | ESK Sequence, ESK.

Encrypted Data
: Symmetrically Encrypted Data Packet | Symmetrically Encrypted and Integrity Protected Data Packet.

Encrypted Message
: Encrypted Data | ESK Sequence, Encrypted Data.

One-Pass Signed Message
: One-Pass Signature Packet, OpenPGP Message, Corresponding Signature Packet.

Signed Message
: Signature Packet, OpenPGP Message | One-Pass Signed Message.

Optionally Padded Message
: OpenPGP Message | OpenPGP Message, Padding Packet.

Кроме того, в любом месте последовательности может присутствовать пакет Marker (параграф 5.8).

10.3.1. Распаковка шифрованных и сжатых сообщений

В дополнение к описанным выше вариантам некоторые сообщения могут быть «распакованы» для получения новых сообщений, в частности:

- расшифровка пакета SEIPD версии 2 **должна** давать действительное сообщение Optionally Padded;
- расшифровка пакета SEIPD версии 1 или SED (старый формат) **должна** давать действительное сообщение;
- декомпрессия пакета Compressed Data **должна** давать действительное сообщение OpenPGP.

При любом разворачивании получаемый поток октетов преобразуется в серию пакетов OpenPGP, как и любой другой поток октетов. Предполагается, что границы пакетов в серии октетов будут соответствовать длине развёрнутого потока октетов. Реализации **недопустимо** интерпретировать октеты на пределами границ развёрнутого потока октетов как часть какого-либо пакета OpenPGP. При обнаружении пакета, где размер заголовка указывает, что он будет выходить за границу развёрнутого потока пакетов реализация **должна** отклонить этот пакет как искажённый и непригодный.

10.3.2. Дополнительные ограничения для последовательностей пакетов

Следует отметить, что некоторые сочетания, разрешённые формально, на деле недопустимы.

10.3.2.1. Версии пакетов в шифрованных сообщениях

Как отмечено выше, Encrypted Message представляет собой последовательность (возможно, пустую) пакетов PKESK (параграф 5.1) и SKESK (параграф 5.3), за которыми следует содержимое (payload) SEIPD (параграф 5.13). В некоторых исторически сохраняющихся случаях содержимым может быть устаревший пакет SED (параграф 5.7) вместо SEIPD, хотя реализациям **недопустимо** генерировать пакеты SED (параграф 13.7). Версии предшествующих пакетов ESK в Encrypted Message **должны** соответствовать версии содержимого пакета SEIPD, как указано здесь.

Пакеты PKESK v3 и SKESK v4 содержат Symmetric Cipher Algorithm ID и сеансовый ключ для последующего пакета SEIPD в обычном виде (cleartext). Поскольку SEIPD v1 не содержит идентификатор симметричного алгоритма, все пакеты ESK, предшествующие содержимому SEIPD v1, **должны** быть PKESK v3 или SKESK v4.

С другой стороны, обычные данные (cleartext) пакетов ESK v6 (PKESK или SKESK) не содержат Symmetric Cipher Algorithm ID, поэтому они не могут сочетаться с содержимым SEIPD v1. Содержимым после любого пакета PKESK v6 или SKESK v6 **должен** быть пакет SEIPD v2.

Кроме того, во избежание возможных конфликтов алгоритмов шифрования и для простоты реализациям **недопустимо** помещать пакеты PKESK v3 или SKESK v4 перед содержимым SEIPD v2.

Версии пакетов, содержащихся в Encrypted Message, показаны в таблице 26. Реализации **должны** генерировать сообщения Encrypted Message с использованием лишь тех версий пакетов, для которых в правом столбце указано «Да». Остальные строки приведены для исторической совместимости. Соблюдающие спецификацию реализации **должны** генерировать Encrypted Message с использованием лишь пакетов, чьи версии указаны в одной строке.

Таблица 26. Реестр версий формата пакетов шифрованных сообщений OpenPGP.

Версия шифрования	***Предшествующий симметричный ключ***	***Версия предшественника***	***Создаётся?***
Данные	Key ESK (если есть)	Public Key ESK (если есть)	
SED (параграф 5.7)	-	PKESK v2 [RFC2440]	Нет
SED (параграф 5.7)	SKESK v4 (параграф 5.3.1)	PKESK v3 (параграф 5.1.1)	Нет
SEIPD v1 (параграф 5.13.1)	SKESK v4 (параграф 5.3.1)	PKESK v3 (параграф 5.1.1)	Да
SEIPD v2 (параграф 5.13.2)	SKESK v6 (параграф 5.3.2)	PKESK v6 (параграф 5.1.2)	Да

Реализации, обрабатывающие Encrypted Message, **должны** отбрасывать любой предшествующий пакет ESK, версия которого не соответствует версии содержимого.

10.3.2.2. Версии пакетов с подписями

Пакеты OpenPGP Key и Signature имеют версию, которая обычно совпадает с версией ключа подписи. Когда ключ версии 6 служит для создания пакета Signature, **должен** создаваться пакет версии 6, независимо от типа пакета Signature. Когда сообщение подписывается или проверяется с применением однопроходной конструкции, версию пакета One-Pass Signature (параграф 5.4) также следует согласовывать с другими версиями.

Некоторые унаследованные реализации создают несогласованные версии старого ключевого материала, которые также указаны в таблице для обеспечения функциональной совместимости. Соответствующая спецификации реализация **должна** создавать пакеты Signature только с номером версии, соответствующим строке со значением «Да» в столбце «Создаётся?».

Таблица 27. Реестр версий ключей и подписей OpenPGP.

Версия ключа подписи	***Версия пакета подписи***	***Версия пакета CPE***	***Создаётся?***
3 (параграф 5.5.2.1)	3 (параграф 5.2.2)	3 (параграф 5.4)	Нет
4 (параграф 5.5.2.2)	3 (параграф 5.2.2)	3 (параграф 5.4)	Нет
4 (параграф 5.5.2.2)	4 (параграф 5.2.3)	3 (параграф 5.4)	Да
6 (параграф 5.5.2.3)	6 (параграф 5.2.3)	6 (параграф 5.4)	Да

Отметим, что несоответствие версий этих пакетов не аннулирует всю последовательность пакетов и лишь делает подпись недействительной, поскольку подпись с неизвестной версией **следует** отбрасывать (параграф 5.2.5).

10.4. Отдельные подписи

Некоторые приложения OpenPGP используют так называемые отдельные подписи (detached signature). Например, пакет программ может включать файл и сопровождающий его файл отдельной подписи. Такая подпись содержит один или несколько пакетов Signature, сохранённых отдельно от подписанных ими данных.

Кроме того, в любом месте последовательности может присутствовать пакет Marker (параграф 5.8) и пакет Padding (параграф 5.14).

11. Криптография с эллиптическими кривыми

В этом разделе описаны алгоритмы и параметры для ключей, используемых в криптографии на основе эллиптических кривых (Elliptic Curve Cryptography или ECC). Базовые сведения о ECC можно найти в работе [KOBLITZ]. В Приложении B.4 к [FIPS186] описаны методы генерации секретных ключей ECC с однородным распределением.

Ни для одного из рассмотренных в документе методов ECC не разрешается применять устаревшие ключи версии 3.

11.1. Кривые ECC

В этом документе упоминаются три именованные кривые над простым полем, заданные в [FIPS186] как Curve P-256, Curve P-384 и Curve P-521, а также три именованные кривые над простым полем, заданные в [RFC5639] как brainpoolP256r1, brainpoolP384r1, brainpoolP512r1. Все 6 кривых могут применяться в алгоритмах с открытым ключом ECDSA и ECDH. Кривые упоминаются с использованием последовательностей октетов, называемых OID. Формирование этих последовательностей октетов подробно описано в параграфе 9.2.

Также заданы отдельные алгоритмы для использования кривых X25519, X448 [RFC7748], Ed25519 и Ed448 [RFC8032]. Дополнительно определены унаследованные OID для Curve25519Legacy (шифрование с использованием алгоритма ECDH) и Ed25519Legacy (подпись с использованием алгоритма EdDSALegacy).

11.2. Форматы передачи точек EC

Точки эллиптической кривой всегда представляются в формате передачи как MPI. Для каждой кривой используется свой формат представления точек внутри MPI при передаче в линию. Каждый формат имеет октет, гарантирующий что в старшем октете установлен хотя бы 1 бит, чтобы значения MPI имели постоянный размер.

Таблица 28. Реестр форматов передачи точек эллиптических кривых OpenPGP.

Имя	*Формат передачи*	*Документ*
SEC1	0x04 || x || y	параграф 11.2.1
Prefixed native	0x40 || native	параграф 11.2.2

11.2.1. Формат передачи точки EC SEC1

Для точки с кодированием SEC1 (без сжатия) MPI имеет вид

```
B = 04 || x || y
```

где x и y - координаты точки P = (x, y), каждая из которых указана в формате big-endian с дополнением нулями до скорректированного размера базового поля, который получается путём округления с увеличением до ближайшего значения, выровненного по 8-битовой границе, как указано в параграфе 9.2 для столбца fsize. Это кодирование совместимо с определением в [SEC1].

11.2.2. Формат передачи естественной точки EC с префиксом

Для пользовательской сжатой точки MPI имеет вид

```
B = 40 || p
```

где p - открытый ключ точки, представленный по правилам кодирования для соответствующей кривой. Этот формат применяется для ключей ECDH на основе кривых в форме Монтгомери и точек при использовании EdDSA.

11.2.3. Замечания по формату передачи точек EC

С учётом приведённых выше определений точный размер содержимого MPI кодированной точки составляет 515 для NIST P-256 и brainpoolP256r1, 771 - для NIST P-384 и brainpoolP384r1, 1059 - для NIST P-521, 1027 - для brainpoolP512r1, и 263 - для Curve25519Legacy и Ed25519Legacy. Например, размер открытого ключа EdDSALegacy для кривой Ed25519Legacy составляет 263 бита - 7 битов представляют октет префикса 0x40 и 32 октета - естественное значение открытого ключа.

При получении нулевой точки (точка на бесконечности) в результате арифметических операций с точкой, её **не следует** включать в структуры данных, определённые в этом документе.

Для каждой конкретной кривой используется заданный формат передачи в линию точки, найденной в её открытом ключе или структуре данных ECDH. Реализациям **недопустимо** использовать для передачи точек форматы, отличающиеся от заданного для кривой формата передачи.

11.3. Форматы передачи скаляра EC

В криптографии с эллиптическими кривыми некоторые значения, не принадлежащие кривой (например, секретные ключи или компоненты подписи), не являются точками кривой, но кодируются в OpenPGP для передачи как MPI.

Из-за различий в схемах развёртывания для некоторых кривых эти значения рассматриваются как неанализируемые (opaque) строки битов, другие - как целые числа в стандартной для OpenPGP форме big-endian. Выбор кодирования зависит от применяемого алгоритма с открытым ключом.

Таблица 29. Реестр кодирования скаляров эллиптических кривых в OpenPGP.

Тип	*Описание*	*Документ*
integer	Целое число в формате big-endian как стандартное значение OpenPGP MPI	параграф 3.2
octet string	Строка октетов фиксированного размера, которая может сокращаться при передаче за счёт вырезания нулей в начале при кодировании MPI с последующим дополнением нулями перед использованием	параграф 11.3.1
prefixed N octets	Строка октетов фиксированного размера N с октетом префикса 0x40 для исключения нуля в начале	параграф 11.3.2

11.3.1. Формат передачи строки октетов EC

Некоторые неанализируемые строки октетов передаются как MPI простым вырезанием нулей в начале с учётом числа оставшихся битов. Такие строки имеют известный фиксированный размер и представляются в документе как MPI(N октетов X), где N - ожидаемое число октетов в строке. Например, 5-октетная строка MPI(5 октетов X), где X имеет значение 00 02 EE 19 00 будет представлена в линии как MPI 00 1A 02 EE 19 00. Для представления X в формате передачи в 2-октетный счётчик битов MPI помещается номер старшего установленного бита (бит 26 или 0x001A), а имеющиеся в начале нули в линию не передаются.

При обратном преобразовании реализация может выполнить указанные ниже действия, если известен ожидаемый размер X (например, 5 октетов):

- убедиться, что значение 2-октетного счётчика битов MPI не превышает 40 (5 октетов по 8 битов);
- выделить 5 октетов, заполненных нулями;
- скопировать октеты данных MPI (без двух октетов счётчика) в младшие позиции выделенного пространства.

11.3.2. Формат передачи строки октетов EC с префиксом

Другим способом обеспечить кодирование строки байтов фиксированного размера для передачи в линию с сохранением формата MPI является добавление к строке префикса в форме специального байта, отличного от 0. Данная спецификация задаёт октет префикса 0x40. Это представляется в документе как MPI(N октетов X с префиксом), где N - известный размер строки байтов.

Например, 5-октетная строка, использующая MPI(5 октетов X с префиксом), где X имеет значение 00 02 EE 19 00, будет передана в линию как 00 2F 40 00 02 EE 19 00. Для кодирования строки к ней добавляется префикс 0x40 (установленный бит 7), затем в 2-октетном счётчике битов MPI устанавливается значение 47 (0x002F - 7 битов префикса и 40 битов строки).

Для декодирования строки из линии реализация, знающая способ формирования переменной, может:

- убедиться, что первые 3 октета MPI (2 октета счётчика битов и октет префикса) имеют значение 00 2F 40;
- использовать оставшуюся часть MPI, полученного из линии.

Отметим, что это похоже на кодирование точек EC, описанное в параграфе 11.2.2.

11.4. Функция вывода ключей

Для шифрования на основе эллиптических кривых необходима функция вывода ключей (KDF). **Требуется** функция вывода ключа с конкатенацией (Concatenation Key Derivation Function, Approved Alternative 1) [SP800-56A] и хэш-функцией KDF SHA2-256 [FIPS180] или более строгой.

Ниже приведён краткий метод кодирования со значительными упрощениями, обусловленными ограниченным выбором функций в этом документе. Нормативное определение содержится в [SP800-56A].

```
//   Реализация KDF( X, oBits, Param );
//   Ввод: точка X = (x,y)
//   oBits - желаемый размер вывода
//   hBits - размер вывода хэш-функции Hash
//   Param - октеты, представляющие параметры
//   Предполагается, что oBits <= hBits
//   Преобразование точки X в строку октетов:
//   ZB' = 04 || x || y
//   Извление x из ZB'
ZB = x;
MB = Hash ( 00 || 00 || 00 || 01 || ZB || Param );
Возврат oBits битов слева в MB.
```

Отметим, что ZB в приведённом описании KDF является компактным представлением X из параграфа 4.2 в [RFC6090].

11.5. Алгоритм ECDH

В этом параграфе описан однопроходный метод DH (One-Pass Diffie-Hellman), который является сочетанием метода ECC DH, создающего общий секрет, и метода вывода ключа, преобразующего этот секрет в ключ. Кроме того, здесь описан метод упаковки ключа, использующий выведенный ключ для защиты сеансового ключа шифрования сообщения.

Однопроходный метод DH C(1, 1, ECC CDH) [SP800-56A] **должен** быть реализован с несколькими ограничениями - примитив сомножителя (ECC Cofactor Diffie-Hellman или CDH) в этом методе изменён так, что для сомножителя всегда принимается значение 1 и применяетчя функция KDF, заданная в параграфе 11.4 с показанными ниже параметрами. Параметры KDF представляются как конкатенация указанных ниже 5 полей переменного и фиксированного размера, совместимых с определением строки битов OtherInfo [SP800-56A].

- Поле переменного размера с OID кривой:
  - 1 октет размера следующего поля;
  - октеты OID кривой в соответствии с параграфом 9.2.
- 1-октетный идентификатор алгоритма с открытым ключом в соответствии с параграфом 9.1.
- Поле переменного размера с параметрами KDF, идентичное соответствующему полю открытого ключа ECDH:
  - 1 октет размера следующих полей (значения 0 и 0xFF зарезервированы для будущих расширений);
  - 1-октетное значение 0x01, зарезервированное для будущих расширений;
  - 1-октетный идентификатор хэш-функции, применяемой с KDF;
  - 1-октетный идентификатор симметричного алгоритма, используемого для упаковки симметричного ключа шифрования сообщения (см. параграф 11.5).
- 20 октетов представления UTF-8 для строки «Anonymous Sender», дополненных в конце пробелами, что даёт в результате последовательность 41 6E 6F 6E 79 6D 6F 75 73 20 53 65 6E 64 65 72 20 20 20 20.
- Поле переменного размера с оттиском субключа шифрования получателя, указывающего требуемый для расшифровки ключевой материал. Для версии 4 поле занимает 20 октетов, для версии 6 - 32.

Размер последовательности параметров KDF в октетах (см. выше) для шифрования с ключом версии 4 составляет 54 для кривой NIST P-256, 51 - для кривых NIST P-384 и NIST P-521, 55 - для brainpoolP256r1, brainpoolP384r1 и brainpoolP512r1 и 56 - для Curve25519Legacy. При шифровании с ключом версии 6 размер последовательности составляет 66 октетов для кривой NIST P-256, 63 - для NIST P-384 и NIST P-521, 67 - для brainpoolP256r1, brainpoolP384r1 и brainpoolP512r1.

Метод упаковки ключа описан в [RFC3394]. KDF создаёт симметричный ключ, используемый в качестве KEK, как указано в [RFC3394]. В параграфе 11.5.1 подробно описан выбор алгоритма KEK, которому **следует** быть одним из трёх

алгоритмов AES. Упаковка и распаковка ключей выполняется с принятым по умолчанию начальным значением из [RFC3394].

Для создания входных данных метода упаковки ключа сначала выполняется конкатенация значений:

- 1-октетного идентификатора алгоритма, если он передан (в случае пакета PKESK v3);
- сеансового ключа;
- 2-октетной контрольной суммы сеансового ключа, равной сумме октетов этого ключа по модулю 65536.

Затем полученная конкатенация дополняется до кратного 8 октетам размера описанным в [RFC8018] методом.

Например, в пакете PKESK версии сеансовый ключ AES-256 кодируется в 40-октетную последовательность

```
09 k0 k1 ... k31 s0 s1 05 05 05 05 05
```

Октеты k0 - k31 указывают сеансовый ключ, s0 и s1 - контрольную сумму октетов сеансового ключа. Такое кодирование позволяет отправителю скрыть размер симметричного ключа, применяемого для шифрования данных. Например, при использовании алгоритма AES для сеансового ключа отправитель **может** использовать 21, 13 или 5 октетов заполнения для AES-128, AES-192 или AES-256, соответственно, чтобы получить 40 октетов на входе метода упаковки ключа.

В пакет PKESK версии 6 симметричный алгоритм не включается, как указано в параграфе 5.1. Например, сеансовый ключ AES-256 будет иметь вид

```
k0 k1 ... k31 s0 s1 06 06 06 06 06 06
```

Октеты k0 - k31 указывают сеансовый ключ, s0 и s1 - контрольную сумму. В этом случае при использовании AES для сеансового ключа отправитель **может** добавить 22, 14 или 6 октетов заполнения для AES-128, AES-192 или AES-256, соответственно, чтобы получить 40 октетов на входе метода упаковки ключа.

Выходной результат метода состоит из двух полей. Первое поле - это MPI эфемерного ключа, использованного для создания общего секрета, второе состоит из двух субполей:

- октет размера результата упаковки ключей в октетах (значение 255 зарезервировано для расширений);
- до 254 октетов, представляющих результат упаковки ключа, применённой к 8-октетному сеансовому ключу с дополнением, как описано выше.

Отметим, что для сеансовых ключей размером 128, 192 и 256 битов результат упаковки ключа имеет размер 32, 40 и 48 октетов, соответственно, если не используется сокрытие (obfuscation) размера.

Для удобства ниже приведено краткое описание метода кодирования, нормативная спецификация приведена в этом разделе, а также в [SP800-56A] и [RFC3394].

- Получение аутентифицированного открытого ключа получателя R.
- Генерация эфемерной одноразовой пары ключей {v, V=vG}.
- Расчёт общей точки S = vR.
- m = symm_alg_ID || сеансовый ключ || контрольная сумма || pkcs5_padding
- curve_OID_len = (octet)len(curve_OID)
- Param = curve_OID_len || curve_OID || public_key_alg_ID || 03 || 01 || KDF_hash_ID || KEK_alg_ID for AESKeyWrap || 41 6E 6F 6E 79 6D 6F 75 73 20 53 65 6E 64 65 72 20 20 20 20 || recipient_fingerprint
- Z_len = размер ключа для KEK_alg_ID, используемого с AESKeyWrap
- Расчёт Z = KDF(S, Z_len, Param)
- Расчёт C = AESKeyWrap(Z, m) (в соответствии с [RFC3394])
- Очистка (затирание) памяти, содержащей S, v и Z для предотвращения утечки эфемерных секретов.
- VB = результат преобразования точки V в строку октетов
- Вывод (MPI(VB) || len(C) || C)

Расшифровка обеспечивается обращением приведённого метода. Отметим, что получатель с ключевой парой (r,R) вычисляет общий секрет как

```
S = rV = rvG
```

11.5.1. Параметры ECDH

Ключи ECDH имеют параметр алгоритма хэширования для вывода ключа и симметричного алгоритма для инкапсуляции ключа.

Для ключей версии 6 **должны** применяться указанные в таблице 30 алгоритмы в соответствии с кривой. Реализации **недопустимо** генерировать ключи ECDH версии 6 для любой из указанных кривых с использованием других параметров KDF или KEK. **Недопустимо** шифровать какое-либо сообщение с ключом ECDH версии 6 для указанной кривой, которая анонсирует иные параметры KDF или KEK.

Для ключей версии 4 следует применять указанные в таблице 30 алгоритмы в соответствии с кривой. Реализации **следует** использовать в качестве алгоритма KEK только AES.

Таблица 30. Реестр параметров OpenPGP ECDH KDF и KEK.

Кривая	***Алгоритм хэширования***	***Симметричный алгоритм***
NIST P-256	SHA2-256	AES-128
NIST P-384	SHA2-384	AES-192
NIST P-521	SHA2-512	AES-256

brainpoolP256r1	SHA2-256	AES-128
brainpoolP384r1	SHA2-384	AES-192
brainpoolP512r1	SHA2-512	AES-256
Curve25519Legacy	SHA2-256	AES-128

12. Замечания по алгоритмам

12.1. Кодирование PKCS#1 в OpenPGP

В этой спецификации применяются функции PKCS#1 EME-PKCS1-v1_5 и EMSA-PKCS1-v1_5, однако соглашения о вызовах для них были изменены в прошлом. Во избежание возможных конфликтов и проблем совместимости в документ включены копии, взятые из PKCS#1 v2.2[1] [RFC8017]. RFC8017 следует считать основным источником PKCS#1 для OpenPGP. Тем не менее, имеет смысл наличие самодостаточного документа, позволяющего избежать проблем в будущем.

12.1.1. EME-PKCS1-v1_5-ENCODE

Вход

k = размер модуля ключа в октетах.
M = сообщение для кодирования в форме строки октетов размером mLen, где mLen <= k - 11.

Выход

EM = закодированное сообщение в форме строки октетов размером k.
Error: «message too long» (сообщение слишком велико).

1. Проверка размера: если mLen > k - 11, вывод сообщения об ошибке message too long и остановка.
2. Генерация строки псевдослучайных ненулевых октетов PS размером k - mLen - 3 (не менее 8 октетов).
3. Конкатенация PS, сообщения M и заполнения для создания кодированного сообщения EM размером k октетов

```
EM = 0x00 || 0x02 || PS || 0x00 || M
```

4. Вывод EM.

12.1.2. EME-PKCS1-v1_5-DECODE

Вход

EM = закодированное сообщение в форме строки октетов.

Выход

M = декодированное сообщение в форме строки октетов.
Error: «decryption error» (ошибка расшифровки).

Для декодирования сообщения EME-PKCS1_v1_5 (EM) оно делится на строку ненулевых октетов PS и сообщение M

```
EM = 0x00 || 0x02 || PS || 0x00 || M
```

Если первый октет EM не имеет шестнадцатеричного значения 0x00, а второй - 0x02, отсутствует октет 0x00, отделяющий PS от M, или размер PS меньше 8 октетов, выводится ошибка «decryption error» и работа останавливается. Различия между ошибками расшифровки и заполнения описаны в параграфе 13.5.

12.1.3. EMSA-PKCS1-v1_5

Этот метод является детерминированным и предусматривает только операцию кодирования.

Вход

Hash = используемая хэш-функция.
M = кодируемое сообщение.
emLen = предусмотренный размер закодированного сообщения в октетах (не менее tLen + 11, где tLen - число октетов DER-представления значения T, рассчитанного в процессе кодирования).

Выход

EM = закодированное сообщение в форме строки октетов размером emLen.
Errors: «message too long» (слишком большое сообщение), «intended encoded message length too short» (предусмотренный размер закодированного сообщения слишком мал).

Этапы

1. К сообщению M применяется хэш-функция для получения значения H:

   ```
   H = Hash(M)
   ```

   Если хэш-функция возвращает ошибку message too long, вывод сообщения message too long и остановка.

2. Пусть T - полный хэш-префикс (Full Hash Prefix) из таблицы 24 для данной хэш-функции, объединённый (конкатенация) с дайджестом H (в форме ASN.1 DER для используемой хеш-функции), а tLen - размер T в октетах.

3. Если emLen < tLen + 11, вывод сообщения «intended encoded message length too short» и остановка.

4. Генерация строки октетов PS размером emLen - tLen - 3 (не менее 8) с шестнадцатеричными значениями 0xFF.

5. Конкатенация PS, хэш-префикса T и другого заполнения для создания закодированного сообщения EM

   ```
   EM = 0x00 || 0x01 || PS || 0x00 || T
   ```

6. Вывод EM.

[1] В оригинале ошибочно указана версия 2.1, см. https://errata.rfc-editor.org/eid9094/. *Прим. перев.*

12.2. Предпочтения для симметричного алгоритма

Предпочтения симметричного алгоритма - это упорядоченный список алгоритмов, воспринимаемых держателем ключа. Список содержится в самоподписи и у держателя ключа может быть несколько разных предпочтений. Например, Алиса может указать для адреса alice@work.com только AES-128, а для alice@home.com - Camellia-256, Twofish и AES-128. Предпочтения могут также указываться в подписи привязки субключа.

Поскольку алгоритм AES-128 **должен** быть реализован, при его отсутствии в списке он неявно принимается в конце. Однако хорошим тоном является его явное указание. Отметим, что реализации, не соответствующие этой спецификации, неявно реализуют только AES-128. Кроме того, реализации предыдущей версии спецификации [RFC4880] в качестве единственного алгоритма, который **должен** быть реализован, имеют TripleDES.

Реализациям **недопустимо** применять симметричный алгоритм, не указанный в списке предпочтений получателя. При шифровании для нескольких получателей реализация находит алгоритм, подходящий для всех получателей. Отметим, что алгоритм AES-128 **должен** быть реализован, поэтому пересечение предпочтений в любом случае непусто.

Если реализация может расшифровать сообщение, которого нет в предпочтениях держателя ключа, ей **следует** расшифровать его, но она **должна** предупредить об этом держателя ключа. Предположим, например, что Алиса (см. выше) использует реализацию с поддержкой всех алгоритмов из этой спецификации, но имеет разные предпочтения для использования на работе и дома. Если ей передадут сообщение с шифром IDEA, не указанным в её предпочтениях, реализация будет выдавать предупреждение об этом, но в идеальном случае Алиса будет способна расшифровать его.

12.2.1. Обычный текст

Алгоритм 0 (plaintext) можно применять лишь для обозначения секретных ключей, хранящихся в открытом виде. Реализациям **недопустимо** применять алгоритм 0 как указанный симметричный шифр для шифрованного пакета данных (параграф 5.7 или 5.13), он пригоден лишь для пакетов Literal Data (параграф 5.9) с нешифрованными литеральными данными.

12.3. Предпочтения для других алгоритмов

Предпочтения для других алгоритмов похожи на предпочтения для симметричных алгоритмов и задают алгоритмы, воспринимаемые держателем ключа. Настройка сжатия и хэширования требуют дополнительного рассмотрения.

12.3.1. Предпочтения для сжатия

Как и в случае с предпочтениями алгоритмов, реализации **недопустимо** использовать сжатие, не указанное в векторе предпочтений. Если метода Uncompressed (0) нет в списке, он неявно помещается в конец списка, т. е. сообщения без сжатия могут передаваться всегда.

Отметим, что в прежних реализациях при отсутствии списка предпочтений для сжатия могло неявно использоваться предпочтение [ZIP(1), Uncompressed(0)] с применением алгоритма ZIP по умолчанию. Поэтому реализации, предпочитающей несжатые данные, **следует** явно указывать это в списке предпочитаемых алгоритмов сжатия.

12.3.1.1. Без сжатия

Алгоритм 0 (без сжатия, uncompressed) может применяться лишь для указания предпочтительного использования несжатых данных. Реализациям **недопустимо** применять алгоритм 0 для пакетов Compressed Data (параграф 5.6), он пригоден лишь в пакетах Literal Data (параграф 5.9) для кодирования несжатых литеральных данных.

12.3.2. Предпочтения для хэширования

Обычно алгоритм выбирает подписывающая, а не проверяющая сторона, поскольку подписывающий редко знает, кто будет проверять его подпись. Предпочтение упрощает согласование протокола на основе цифровых подписей. Если Алиса аутентифицирует себя Бобу с помощью подписи, ей логично выбрать алгоритм, применяемый в реализации Боба. Предпочтение позволяет Бобу указать, какие алгоритмы Алиса может использовать в своём ключе.

Поскольку SHA2-256 является алгоритмом, которых **должен** быть реализован, при его отсутствии в списке предпочтений он неявно предполагается в конце списка. Однако хорошим тоном является явное указание алгоритма.

12.4. RSA

Схема заполнения PKCS1-v1_5, применяемая в алгоритмах RSA, заданных в этом документе, больше не является рекомендуемой и отменена [SP800-131A]. Поэтому реализациям **не следует** генерировать ключи RSA.

Имеются типы алгоритмов только для ключей подписей (RSA Sign-Only) и только для ключей шифрования (RSA Encrypt-Only). От этих типов отказались в пользу субпакета подписи Key Flags. Реализациям **недопустимо** создавать такие ключи, но **можно** интерпретировать их.

Реализациям **недопустимо** генерировать ключи RSA размером меньше 3072 битов и **не следует** применять такие ключи для шифрования, подписи или проверки. Реализациям **недопустимо** применять для шифрования, подписи или проверки ключи RSA размером меньше 2048 битов. Реализации, расшифровывающей сообщение с использованием секретного ключа RSA размером меньше 3072 битов, **следует** выдавать предупреждение о ненадёжности ключа для современного применения.

12.5. DSA

DSA больше не является рекомендуемым и отменен [FIPS186], поэтому реализациям **недопустимо** генерировать ключи DSA. Реализациям **недопустимо** применять ключи DSA для подписей и их проверки.

12.6. Elgamal

Схема заполнения PKCS1-v1_5, применяемая в алгоритме Elgamal, заданном в этом документе, больше не является рекомендуемой и отменена [SP800-131A]. Поэтому реализациям **недопустимо** генерировать ключи Elgamal.

Реализациям **недопустимо** применять ключи Elgamal для шифрования. Реализации, расшифровывающей сообщение с использованием секретного ключа Elgamal, **следует** выдавать предупреждение о ненадёжности ключа для современного применения.

12.7. EdDSA

Хотя алгоритм EdDSA принимает на входе произвольные данные, при его использовании с OpenPGP на вход должен подаваться дайджест сообщения (предварительное хэширование), как указано в параграфе 5.2.4. Отсечка для дайджеста не применяется и его значение в неизменном виде подаётся на вход алгоритма EdDSA.

Хотя в [RFC8032] описаны разные варианты EdDSA, в OpenPGP применяется лишь «чистый» вариант (PureEdDSA). Хэширование в OpenPGP выполняется как часть стандартного процесса создания подписи OpenPGP, а сам хэш служит входным сообщением для алгоритма PureEdDSA.

Как указано в [RFC8032], для Ed448 нужна «строка контекста», которая в OpenPGP для Ed448 пуста.

12.8. Идентификаторы резервных алгоритмов

Ряд идентификаторов зарезервирован для алгоритмов, которые были бы полезны в реализациях OpenPGP, однако имеются препятствия их внедрению. Такие алгоритмы указаны в параграфе 9.1 как резервные.

Зарезервированный алгоритм с открытым ключом X9.42 (21) не имеет необходимых параметров, их порядка и семантики. Такая же ситуация наблюдается для зарезервированных алгоритмов AEDH (23) и AEDSA (24).

Прежние версии OpenPGP разрешали подписи [ELGAMAL] с открытым ключом по алгоритму 20, которые больше не разрешены, и реализациям **недопустимо** генерировать такие ключи, а также **недопустимо** создавать подписи Elgamal (см. [BLEICHENBACHER]).

12.9. Режим CFB

Режим шифрования с обратной связью (Cipher Feedback или CFB), используемый в этом документе, задан в параграфе 6.3 [SP800-38A]. Размер сегмента CFB s равен размеру блока в шифре (т. е. в n-битовом режиме CFB n указывает размер блока).

12.10. Приватные и экспериментальные параметры

Спецификаторы S2K, идентификаторы типа субпакетов подписи (Signature Subpacket Type ID), субпакетов атрибутов пользователя (User Attribute Subpacket Type ID), формата изображений и другие идентификаторы алгоритмов, описанные в разделе 9, относятся к резервному диапазону 100 - 110 для приватного использования и экспериментов (Private and Experimental Use). Для идентификаторов типа пакетов (Packet Type ID) зарезервированы значения 60 - 63 из того же диапазона (Private and Experimental Use). Выделение значений из этого диапазона происходит по процедуре Private Use and Experimental Use, описанной в [RFC8126].

Следует соблюдать осторожность с такими значениями и переводить их в разряд управляемых IANA параметров при расширении области применения.

12.11. Соображения по части расширения

При расширении OpenPGP с потерей совместимости с прежними версиями (старые реализации не могут корректно работать с новой функцией), предложение о расширении может быть заявлено в самоподписи держателя ключа как часть субпакета подписи Features.

Невозможно с уверенностью сказать, какие расширения не будут совместимы с будущими версиями, но обычно новые алгоритмы совместимы, а новые пакеты - нет.

Если предложение о расширении не обновляет систему Features, в него **следует** включать объяснение причин отсутствия такой необходимости. Если предложение о расширении не содержит ни расширения системы Features, ни объяснения причин отсутствия необходимости, его **следует** отклонять.

13. Вопросы безопасности

- Как и в случае других технологий, включающих криптографию, разработчикам нужно ознакомиться с современной литературой для выяснения возможных уязвимостей используемых алгоритмов. При наличии таких уязвимостей разработчикам следует рассмотреть возможности запрета соответствующих алгоритмов для новых данных и предупреждения конечных пользователей или предотвращения попыток использовать защищённые такими алгоритмами данные.
- В этой спецификации используются технологии криптографии с открытым ключом. Предполагается, что секретный ключ пары находится под контролем и защитой соответствующей стороны или сторон.
- В алгоритмах хэширования MD5 и SHA-1 обнаружены слабые места, а в ряде случаев возникают коллизии. Использование MD5 и SHA-1 в OpenPGP отменено (параграф 9.5).
- Многие разработчики протоколов безопасности считают неразумным использование одного ключа для защиты конфиденциальности (шифрование) и целостности (подписи). Это стало одним из мотивов разработки формата ключей версии 4, где для подписи и шифрования служат разные ключи. Использовать один ключ для подписи и шифрования не рекомендуется.
- Алгоритм DSA работает с любым хэшем, но чувствителен к качеству хэширования. Проверяющим следует знать, что даже при использовании для подписи строгой хэш-функции, злоумышленник может поменять

подпись для использования более слабого хэша. Действительными следует считать лишь подписи с достаточно строгими алгоритмами хэширования.

- Поскольку OpenPGP сочетает множество асимметричных и симметричных алгоритмов, а также алгоритмов хэширования с разной степенью надёжности, следует позаботиться о том, чтобы самый слабый элемент в сообщении OpenPGP оставался достаточно надёжным для достижения поставленной цели. Хотя представление о строгости алгоритма может меняться, в NIST Special Publication 800-57 [SP800-57] даны рекомендации (актуальные на момент публикации) по сравнению уровней безопасности разных алгоритмов.

- С подписями связана возможная проблема безопасности. Если злоумышленник сможет найти сообщение с таким же хэш-значением, полученным по иному алгоритму, он сможет создать поддельную структуру подписи, которая пройдёт проверку. Предположим, например, что Алиса подписала сообщение M с помощью DSA, используя алгоритм хэширования H, а Мэллет находит сообщение M', имеющее такое же хэш-значение, как у M, используя алгоритм H'. Тогда Мэллет может создать блок подписи, который пройдёт проверку как подпись Алисы для сообщения M' с алгоритмом H'. Однако это тоже представляет слабость H, H' или обоих алгоритмов. Если это произойдёт, потребуется пересмотреть этот документ в части разрешённых алгоритмов хэширования.

- При создании системы проверки подлинности получатель может задать предпочтительный алгоритм подписи, однако подписывающему будет неразумно использовать более слабый алгоритм лишь потому, что его запросил получатель.

- Некоторые из упомянутых в документе алгоритмов шифрования проанализированы меньше, чем другие. Например, алгоритм TWOFISH в настоящее время считается достаточно строгим, однако его не анализировали так же тщательно, как AES. С другими алгоритмами могут быть связаны иные опасения.

- В конце лета 2002 г. Джалад (Jallad), Кац (Katz) и Шнайер (Schneier) опубликовали описание интересной атаки на предыдущие версии спецификации OpenPGP и некоторые из реализаций этой атаки [JKS02]. Здесь атакующий меняет сообщение и передаёт его пользователю, который возвращает ошибочно расшифрованное сообщение. Таким образом, пользователь может послужить оракулом для злоумышленника, зачастую корректно расшифровывая сообщения. Это является особой формой атаки с подменой шифротекста. В параграфе 13.7 рассмотрена защита от таких атак в новых версиях OpenPGP.

13.1. Обнаружение коллизий SHA-1

Как указано в [SHAMBLES], алгоритм хэширования SHA-1 не стоек к коллизиям. Однако в реализации OpenPGP нет возможности полного отказа от SHA-1, поскольку этот алгоритм нужен для открытых ключей версии 4. В частности, SHA-1 применяется при создании цифровых оттисков версии 4. Поэтому при поддержке реализацией OpenPGP открытых ключей версии 4 она должна реализовать SHA-1 по меньшей мере в некоторых случаях.

Во избежание риска отказов из-за злонамеренного внедрения коллизий SHA-1 реализация OpenPGP **может** попытаться обнаружить входные данные хэш-функции, полученные в результате известной атаки с коллизией, и отвергнуть такие данные или изменить результат хэширования. Этому действию следует переводить неопределённый сбой (когда неясен эффект коллизии) в явный отказ, что более приемлемо для отказоустойчивой реализации.

В [STEVENS2013] описан метод обнаружения признаков известных атак с коллизиями в SHA-1. Примеры реализации этого метода на языке C представлены в [SHA1CD].

13.2. Преимущества подписей с затравкой

В подписи версии 6 включается хэшируемая первой затравка (salt), размер которой зависит от алгоритма хэширования. Это делает подписи OpenPGP версии 6 недетерминированными и защищёнными от широкого спектра атак, основанных на создании подписи для предсказуемого сообщения. Выбор новой случайной затравки для каждой создаваемой подписи и подписанного хэша делает подписи непредсказуемыми.

Хотя подписываемый материал может контролироваться злоумышленником, хэширование затравки в первую очередь обеспечивает отсутствие контролируемого злоумышленником префикса. Пример таких атак описан в статье «SHA-1 is a Shambles» [SHAMBLES], где рассматривается атака на SHA-1 с коллизией при выбранном префиксе. Злоумышленник, организовавший атаку с выбранным сообщением, не сможет контролировать подписываемый хэш и для создания двух сообщений с одинаковыми подписями ему потребуется нарушить стойкость к поиску второго прообраза вместо более простого нарушения стойкости к коллизиям. Размер затравки связан с хэш-функцией, чтобы обеспечить ожидаемый уровень стойкости к коллизиям, и составляет не менее 16 октетов.

Иногда атакующий может вынудить к созданию подписи, даже не контролируя содержимое сообщения. В отдельных случаях повторная подпись того же сообщения может приводить к утечке части или всего ключа подписи. Пример этого представлен в обсуждении аппаратных отказов EdDSA и детерминированном алгоритме ECDSA в [PSSLR17]. Выбор новой случайной затравки для каждой подписи обеспечивает неповторимость подписей, что снижает риск.

13.3. Побочные каналы эллиптических кривых

Атаки по побочным каналам создают проблему, когда использование совместимым приложением формата OpenPGP может быть смоделировано с помощью оракула расшифровки или подписания. Примером может служить приложение сетевой службы, выполняющей расшифровку данных для удалённых пользователей без аутентификации. Операции скалярного умножения в ECC, применяемые алгоритмами ECDSA и ECDH, уязвимы к атакам по побочным каналам. Зачастую возможны меры противодействия на вышележащем уровне, такие как ограничение числа допустимых отказов и маскировка времени выполнения операций, связанных с каждым интерфейсом. Меры смягчения атак на уровне скалярного умножения направлены на устранение любых измеряемых отличий между операциями сложения и удвоения для операций с точками ECC.

13.4. Риски, связанные с быстрой проверкой

Зимой 2005 г. Серж Мистер (Serge Mister) и Роберт Цуккерато (Robert Zuccherato) из Entrust опубликовали статью описывающую способ, с помощью которого «быструю проверку» пакетов SEIPD и SED версии 1 можно использовать в

качестве оракула для расшифровки двух октетов каждого блока шифра [MZ05]. Проверка была предназначена для раннего обнаружения ошибок расшифровки сеансового ключа, в частности, неверной парольной фразы, поскольку пакеты v4 SKESK не включают проверку целостности.

При использовании быстрой проверки возникает опасность того, что сведения о времени или ошибках при проверке могут быть раскрыты злоумышленником, особенно через автоматизированную службу, позволяющую быстро повторять запросы. Запрет быстрой проверки предотвращает такие атаки.

Для очень больших объёмов шифрованных данных, где сеансовый ключ защищён парольной фразой с использованием v4 SKESK, быстрая проверка может быть удобна для пользователя, позволяя заранее информировать его о вводе ошибочного пароля. Однако реализациям следует применять быструю проверку с осторожностью. Для безопасного и быстрого обнаружения отказов при расшифровке рекомендуется шифровать данные с использованием v2 SEIPD. Если сеансовый ключ зашифрован с открытым ключом, быстрая проверка бесполезна, поскольку шифрование сеансового ключа с открытым ключом должно гарантировать, что это именно тот сеансовый ключ.

Атаки с оракулом быстрой проверки являются частным случаем атак с подменой шифротекста. Сведения о других похожих атаках приведены в параграфе 13.7.

13.5. Предотвращение ошибок из-за PKCS#1

В дополнении PKCS#1 (применяется в PKESK с шифрованием RSA и ElGamal) обнаружена уязвимость к атакам, при которых система, позволяющая отличать ошибки дополнения от других ошибок расшифровки, может выступать как оракул расшифровки и/или подписания, что может приводить к утечке сеансового ключа или разрешать подписывать произвольные данные [BLEICHENBACHER-PKCS1]. Число запросов, требуемых для выполнения атаки может составлять тысячи и миллионы в зависимости от строгости и осторожности реализации обработки дополнения. Для усложнения таких атак реализациям **следует** поддерживать строгую и отказоустойчивую проверку дополнения за постоянное время.

Для предотвращения атак в случаях, когда у злоумышленника нет доступа к данным о времени расшифровки, простейшим решением является выдача одного и того же кода для всех вариантов ошибок при обработке PKESK, а также ошибок целостности SEIPD (включая ошибки разбора сеансового ключа, такие как непригодный алгоритм шифрования для v3 PKESK или несоответствие сеансового ключа для v6 PKESK). Если у атакующего есть доступ к данным о времени, требуется обеспечить постоянство времени обработки. Это требует тщательной проработки, особенно для v3 PKESK, где размер сеансового ключа и сведения о шифре не известны заранее, поскольку являются частью зашифрованных данных PKESK.

13.6. Применимость оттисков

В спецификации применяются оттиски (fingerprint) в нескольких местах при передаче (например, в параграфах 5.2.3.23, 5.2.3.35, 5.2.3.36) и обработке (например, в ECDH KDF, параграф 11.5). Реализации могут применять оттиски по своему усмотрению, например, как индекс хранилища ключей.

Некоторые пользователи OpenPGP использовали сравнение оттисков вручную для проверки открытого ключа партнёра. В версии 4 оттиски обычно представлялись 40 шестнадцатеричными цифрами, сгруппированными по 4 с разделением пробелами. Проверка такой последовательности с участием человека вызывает сомнения. Маловероятно, что многие способны точно сравнить данные с высокой энтропией, особенно при их представлении в компактной текстовой форме, такой как шестнадцатеричное значение [USENIX-STUDY]. В версии 6 сравнение оттисков человеком ещё усложнилось, поскольку размер оттиска вырос до 256 битов (160 в версии 4).

В реализациях OpenPGP следует отдавать предпочтение механическому переносу и сравнению и **не следует** поощрять перенос вручную и сравнение полных оттисков человеком при наличии иных способов.

Хотя в этом разделе упоминается практика представления оттисков версии 4 для человека, документ не пытается стандартизировать какую-либо форму представления человеку оттисков версии 6 для такого нежелательного применения.

> Примечание. Тема проверки ключей с участием человек требует отдельного рассмотрения, которое будет представлено в отдельном документе.

13.7. Предотвращение подделки шифротекста

Если шифротекст может быть изменён злоумышленником, но потом все же расшифрован как некий открытый текст, он называется податливым (malleable). На любую систему с податливым шифрованием можно организовать множество атак, поэтому в [RFC4880] и более поздних версиях OpenPGP имеются механизмы защиты от таких атак. Однако данные OpenPGP могут оказаться созданными до появления таких механизмов. Поскольку реализации OpenPGP работают с данными, которые могли быть созданы давно, они могут столкнуться с податливыми шифротекстами.

Когда реализация OpenPGP обнаруживает, что она расшифровывает данные, которые могут быть податливыми, она **должна** сгенерировать чёткое сообщение об ошибке, указывающее сомнительную целостность данных, ей **не следует** пытаться анализировать или передавать расшифрованные данные пользователю, но **следует** прекратить работу с возвратом ошибки. Анализ или публикация расшифрованных данных до подтверждения их целостности может приводить к утечке расшифрованных данных [EFAIL] [MRLG15].

Если для данных, зашифрованных AEAD, тег аутентификации не проходит проверку, реализации **недопустимо** пытаться анализировать данные или предоставлять их пользователю и она **должна** прекратить работу с возвратом ошибки.

Реализация, встретившая податливый шифротекст, **может** передать его пользователю, если он не был зашифрован с применением AEAD (это говорит о возможной обработке давних данных) и пользователь осведомлён о рисках.

Если сообщение с шифрованием AEAD усечено, т. е. отсутствует последний фрагмент (chunk) с нулевым октетом и, возможно, (частично) отсутствуют некоторые фрагменты перед ним, реализация **может** передать пользователю

расшифрованный текст из полностью аутентифицированных фрагментов при работе в потоковом режиме, но **должна** чётко указать ошибку после обнаружения отсечки.

Ниже перечислены элементы данных OpenPGP, указывающие наличие податливого шифротекста:

- все пакеты Symmetrically Encrypted Data (параграф 5.7);
- любой пакет Compressed Data внутри шифрованного контейнера (параграф 5.6) при отказе расшифровки;
- любой пакет SEIPD версии 1 (параграф 5.13.1), где внутренний код обнаружения изменений (Modification Detection Code) не проходит проверку;
- любой пакет SEIPD версии 2 (параграф 5.13.2), где тег аутентификации любого фрагмента не проходит проверку или отсутствует конечный фрагмент с нулевым октетом;
- любой пакет Secret-Key с зашифрованным ключевым материалом (параграф 3.7.2.1), где не прошла проверка целостности на основе октета защиты секретного ключа:
  - значение 253 (AEAD) указывает недействительный тег аутентификации AEAD;
  - значение 254 (CFB) указывает несоответствие контрольной суммы SHA1;
  - значение 255 (MalleableCFB) или необработанный (raw) алгоритм шифрования, если 2-октетная контрольная сумма в трейлере не совпадает.

Для предотвращения таких случаев реализации, создающей шифрованные данные OpenPGP, **следует** выбирать формат шифрованного контейнера с наиболее надёжной защитой, которую могут обработать предполагаемые получатели. В частности:

- **недопустимо** генерировать устаревшие пакеты SED;
- при шифровании с одним или несколькими открытыми ключами:
  - если все ключи получателей указывают поддержку пакетов SEIPD версии 2 в своём субпакет подписи Features (параграф 5.2.3.32), являются ключами версии 6 без субпакета подписи Features или реализация может иным способом определить, что все получатели поддерживают пакеты v2 SEIPD, **следует** выполнять шифрование с использованием пакета v2 SEIPD;
  - если один из получателей не поддерживает пакеты v2 SEIPD, создатель сообщения **может** использовать взамен пакет v1 SEIPD;
- защищённый паролем секретный ключевой материал в пакете версии 6 Secret Key или Secret Subkey **следует** защищать шифрованием AEAD (октет использования S2K 253), если он не передаётся реализации, заведомо не поддерживающей AEAD; реализации следует осознавать, что в случаях, когда атакующий имеет возможность записи для нешифрованных сеансовых ключей, ключи с шифрованием CFB (октет использования S2K 254 или 255) уязвимы к атакам с повреждением, которые могут приводить к утечке секретных данных при использовании секретного ключа [KOPENPGP] [KR02].

Разработчикам следует незамедлительно реализовать AEAD (v2 SEIPD и октет использования S2K 253) и способствовать его распространению.

Пользователям **рекомендуется** перейти на AEAD.

13.8. Безопасное использование свойства v2 SEIPD Session-Key-Reuse

Вывод ключей пакета v2 SEIPD (параграф 5.13.2) с затравкой (salt) позволяет получателю шифрованного пакета ответить отправителю и всем другим получателям без их открытых ключей, используя те же пакеты v6 PKESK, которые были получены, и другое случайное значение salt. Это обеспечивает на криптографическом уровне безопасный механизм, позволяющий шифровать сообщения в случае отсутствия у отправителя пригодного для шифрования сертификата для одного или нескольких участников взаимодействия, что повышает общий уровень безопасности приложения. Однако при использовании механизма требуется соблюдать осторожность, чтобы не возникло уязвимостей, указанных ниже.

- Ответ лишь части из исходных получателей и отправителя с помощью функции повторного использования сеансового ключа (session-key-reuse) будет означать, что остальные получатели и отправитель исходного сообщения тоже смогут прочесть зашифрованный отклик.
- Добавление получателя в отклик, шифруемый с помощью функции session-key-reuse, предоставляет этому получателю криптографический доступ к сообщению, на которое был передан ответ (возможно и к предшествующим сообщениям).
- Описанное выше изменение списка получателей требует защиты, если сообщение исходно было создано как отклик с сеансовым ключом, затем сохранено (например, как черновик) и снова открыто для редактирования и последующей отправки.
- Имеется потенциальная угроза того, что злоумышленник с доступом к сети или почтовому ящику, являющийся в то же время получателем исходного сообщения, может незаметно удалить себя из сообщения до получения сообщения клиентом жертвы. Этот клиент может затем использовать механизм отклика с повторным использованием сеансового ключа, создав зашифрованное сообщение, которое будет доступно злоумышленнику. Реализациям рекомендуется использовать субпакет Intended Recipient Fingerprint (параграф 5.2.3.36) при создании сообщений и проверке согласованности списка получателей сообщения перед отправкой отклика с повторным использованием сеансового ключа.
- При использовании функции session-key-reuse в любом протоколе вышележащего уровня нужно убедиться, что в этом протоколе нет препятствий повторному использованию сеансовых ключей. Такие помехи могут возникать, например, (если исходное сообщение уже составлено) из-за повторного использования сеансового

ключа из имеющегося зашифрованного файла, который уже мог стать общим для группы пользователей. Использование функции session-key-reuse для составления шифрованного отклика на такое сообщение может приводить к тому, что вся группа получит криптографический доступ к зашифрованному сообщению.

- Как правило, контроль использования функции session-key-reuse следует отдавать пользователю. В частности, следует проявлять осторожность, чтобы функция не применялась незаметно, когда пользователь считает, что для шифрования используется подобающий открытый ключ. Это может происходить, например, в случае, когда открытый ключ одного из получателей известен, но срок его действия истёк. Следует проявлять особую осторожность, чтобы пользователи не продолжали неосознанно применять сеансовые ключи повторно, а как можно скорее переключались бы на использование свежего открытого ключа шифрования.
- По возможности клиенту следует предпочитать шифрование со свежим открытым ключом повторному использованию сеансового ключа.

Хотя это может и не быть связано с аспектами безопасности, следует отметить, что исходное составление зашифрованного ответа с использованием функции session-key-reuse у одного клиента с последующим сохранением (например, как черновика) и повторным открытием сохранённого незавершённого ответа другим клиентом, который не поддерживает функцию повторного использования сеансового ключа, может приводить к проблемам совместимости.

Для предотвращения описанных выше сложностей требуются познания в связанной с контекстом области. Реализации следует применять функцию session-key-reuse лишь на том уровне конкретного приложения, где это возможно в соответствии с обоснованными рекомендациями по безопасному использованию функции в таком приложении. На момент создания этого документа не было известно рекомендаций по безопасному повторному использованию сеансовых ключей OpenPGP для какого-либо конкретного контекста. Планирующим использовать эту функцию разработчикам следует опубликовать свои рекомендации для других.

13.9. Депонированные подписи отзывов

Алиса (держатель ключа) может назначить другое лицо, которое может отозвать её ключ. Предпочтительным способом для этого является создание специальной подписи отзыва (Revocation Signature, Signature Type ID 0x20, 0x28 или 0x30) и её безопасная передача выбранному лицу (органу) для помещения в хранилище депонирования. Выбранное лицо может опубликовать депонированную подпись отзыва в любое удобное время вместо генерации Revocation Signature.

Ниже указаны преимущества использования депонированной подписи отзыва по сравнению с устаревшим субпакетом Revocation Key (параграф 5.2.3.23).

- Держатель ключа может ограничить типы отзыва только конкретными депонированными подписями.
- Между держателем ключа и отзывающим нет публичной/видимой связи.
- Отзыв доступен для проверки третьим лицам (сторонам) без поиска ключа отзывающего.
- Отзывающий не обязан иметь открытый ключ OpenPGP при наличии между ним и держателем ключа иного защищённого транспорта.
- Отзыв с использованием субпакета Revocation Key не обеспечивает чёткости и надёжности.
- Наличие криптографической уязвимости в механизме оттиска (fingerprint) не влияет на депонированные Revocation Signature.

Revocation Signature можно разделить и передать нескольким лицам (органам), чтобы восстановить подпись отзыва они могли лишь совместными действиями.

13.10. Генерация случайных значений и затравки

Для OpenPGP нужен криптографически безопасный генератор псевдослучайных чисел (cryptographically secure pseudorandom number generator или CSPRNG). В большинстве случаев операционная система предоставляет подходящий инструмент (например, системный вызов getrandom() в Linux и BSD), который следует применять, если нет других требований (например, по производительности). **Рекомендуется** использовать имеющуюся реализацию CSPRNG, а не создавать новую. Имеется множество криптографических библиотек с благоприятным лицензированием. Если они не устраивают, следует обратиться к [RFC4086], где даны советы по генерации случайных чисел.

OpenPGP использует случайные данные с тремя уровнями видимости:

- общедоступные значения, такие как nonce, векторы инициализации (IV), открытый материал для заполнения, затравки (salt);
- общие секретные значения, такие как сеансовые ключи для шифрованных данных и материал для заполнения шифрованных пакетов;
- полностью секретные данные, такие генерируемые асимметричные ключи.

При корректной работе CSPRNG разные варианты видимости не вызывают проблем безопасности, поскольку по выходным данным CSPRNG невозможно определить состояние генератора. Однако при взломе CSPRNG атакующий может получить возможность определять внутреннее состояние генератора по его выводу и предсказывать менее видимые данные, такие как ключевой материал (см. [CHECKOWAY]). Реализации могут обеспечивать дополнительную защиту от таких атак, используя разные CSPRNG для генерации случайных данных с разным уровнем видимости.

13.11. Анализ трафика

При передаче данных OpenPGP через сеть размер этих данных может стать источником утечки информации злоумышленнику. В некоторых ситуациях такая утечка недопустима с точки зрения безопасности. Например, если известно, что сообщения данного протокола в открытом виде имеют значение yes (3 октета) или no (2 октета), и передаются в пакетах SEIPD, размер шифрованных пакетов будет раскрывать содержимое открытых данных.

Другим примером является передача OpenPGP Transferable Public Key через сетевое соединение с шифрованием, где может раскрываться размер сертификата. Поскольку размер сертификата OpenPGP зависит от содержимого, внешний наблюдатель, заинтересованный в метаданных (например, человек, пытающийся связаться с другим человеком), может угадать отождествление в переданном сертификате, если его размер уникален.

В обоих случаях реализация может изменить размер составной структуры, включая пакет Padding (параграф 5.14).

13.12. Скрытая пересылка

При получении злоумышленником подписи для какого-либо текста (например, приём сообщения с подписью) он может использовать эту подпись в своих целях, передав другому получателю сообщение с тем же содержимым и подписью якобы от имени исходного отправителя. Для предотвращения этого реализациям **следует** поддерживать субпакеты Intended Recipient Fingerprint (параграф 5.2.3.36).

13.13. Хэшированные и нехэшированные субпакеты

Каждая подпись OpenPGP может иметь субпакеты в двух разных разделах. Первый набор субпакетов (хэшируемая секция) учитывается при создании подписи, а второй не имеет криптографической защиты и применяется только для вспомогательных данных, включая локально сохранённые аннотации к подписи. Рассмотрим, например, реализацию, работающую с конкретной подписью, о которой известно, что она была создана с определенным ключом, хотя в хэшированном разделе подписи нет субпакета Issuer Fingerprint (параграф 5.2.3.35). Такая реализация **может** синтезировать пакет Issuer Fingerprint и сохранить его в нехэшируемой секции, чтобы в будущем иметь возможность определить ключ, использованный для создания подписи.

Некоторые субпакеты полезны лишь в хэшированном разделе и реализации **следует** игнорировать такие субпакеты если они найдены в нехэшированном разделе и источник не известен. Например, субпакет Preferred AEAD Ciphersuites (параграф 5.2.3.15) в самоподписи Direct Key указывает предпочтения держателя ключа при шифровании данных v2 SEIPD с этим ключом. Реализация, учитывающая наличие этого субпакета в нехэшированном разделе, будет уязвима для атаки с подделкой сертификата получателя с целью вынудить к использованию определённого шифра или режима работы.

13.14. Вредоносные сжатые данные

Можно создать сжатый куайн[1] (quine), воспроизводящий себя после декомпрессии, что приведёт к бесконечной регрессии в любой реализации, которая готова анализировать произвольное число уровней сжатия. Это может истощить ресурсы, что, в свою очередь, может привести к отказу операционной системы. Если операционная система создаст отчёт об отказе (crash report), в нем могут оказаться конфиденциальные сведения.

Реализации OpenPGP **следует** ограничивать число уровней сжатия, для которых она готова выполнить декомпрессию в одном сообщении.

14. Вопросы реализации

В этом разделе приведены комментарии в помощь разработчикам, заинтересованным в совместимости с прежними версиями. Различия зачастую невелики, но иной раз доставляют больше хлопот, нежели крупные изменения. Ниже указан краткий список возможных проблем и подводных камней, с которыми могут столкнуться разработчики.

- Имеется много вариантов, при которых два ключа имеют одинаковый материал, но разные оттиски (и разные Key ID). Например, в версии 4 оттиск создаётся путём хэширования времени создания ключа, наряду с другими данными, поэтому два ключа версии 4, созданные на одном материале, но в разное время, будут иметь разные оттиски.
- OpenPGP не ограничивает размер открытых ключей, однако большой ключ не обязательно будет лучше. При использовании больших ключей растёт время расчётов и они могут стать непрактичными. В разных реализациях OpenPGP могут присутствовать свои верхние границы размера открытых ключей, поэтому для совместимости следует осторожно выбирать размер ключей.
- ASCII Armor является необязательной функцией OpenPGP. Рабочая группа OpenPGP стремится минимизировать набор обязательных для реализации функций, поэтому возможны реализации, применяющие только двоичные форматы объектов. Например, реализация, применяющая OpenPGP как механизм подписи файлов, может не нуждаться в ASCII Armor. OpenPGP разрешает реализациям объявлять поддерживаемые и неподдерживаемые функции, но ASCII Armor не входит в их число. Поскольку большинство реализаций разрешает использовать двоичные и защищённые (armored) объекты без разбора, реализация без поддержки ASCII Armor может столкнуться с проблемами совместимости с реализациями общего назначения. Более того, реализации OpenPGP-MIME [RFC3156] уже требуют ASCII Armor, поэтому они включают такую поддержку.
- То, что в этом документе называется унаследованным (legacy) форматом пакетов (параграф 4.2.2), в прежних версиях называлось старым форматом, который применялся в реализациях, предшествующих [RFC2440]. Текущий формат пакетов OpenPGP (параграф 4.2.1) в более старых RFC назывался новым. Этот формат задан в [RFC2440] и поддерживается в [RFC4880] и данном документе.

14.1. Хранилище унаследованных оттисков ключей версии 6 с ограничениями

Некоторые реализации OpenPGP имеют фиксированные ограничения размера хранимых оттисков ключей из-за чего не помещаются все 32 октета оттиска версии 6. Например, в [OPENPGPCARD] для каждого оттиска разрешены 20 октетов.

Реализации OpenPGP **недопустимо** сопоставлять какую-либо часть оттиска версии 6 с таким ограниченным полем, если только в спецификации среды с ограничением явно не указаны рекомендации по сохранению оттисков версии 6, позволяющие отличить их от оттисков версии 4. Реализации, взаимодействующей с таким ограниченным полем, **следует** напрямую вычислять оттиск версии 6 по открытому ключу и связанным метаданным, не полагаясь на поле с ограничением.

[1] Программа, выводящая свой исходный код. *Прим. перев.*

15. Взаимодействие с IANA

Этот документ отменяет [RFC4880]. Агентство IANA обновило все регистрационные данные, ссылавшиеся на [RFC4880], с указанием ссылок на этот RFC.

15.1. Переименованная группа протоколов

IANA собирает реестры, связанные с определенным протоколом, в «группу протокола» и название группы Pretty Good Privacy (PGP) (группа реестров <https://www.iana.org/assignments/pgp-parameters>) заменено на OpenPGP с постоянной переадресацией имеющегося URL на URL новой группы. Все указанные ниже обновления относятся к реестрам группы OpenPGP.

15.2. Переименованные и обновлённые реестры

Реестр PGP String-to-Key (S2K) переименован в OpenPGP String-to-Key (S2K) Types и его содержимое обновлено в соответствии с таблицей 1.

Реестр PGP Packet Types/Tags переименован в OpenPGP Packet Types и его содержимое обновлено в соответствии с таблицей 3.

Реестр Signature Subpacket Types переименован в OpenPGP Signature Subpacket Types и его содержимое обновлено в соответствии с таблицей 5.

Реестр Key Server Preference Extensions переименован в OpenPGP Key Server Preference Flags и его содержимое обновлено в соответствии с таблицей 8.

Реестр Key Flags Extensions переименован в OpenPGP Key Flags и его содержимое обновлено в соответствии с таблицей 9.

Реестр Reason for Revocation Extensions переименован в OpenPGP Reason for Revocation (Revocation Octet) и его содержимое обновлено в соответствии с таблицей 10.

Реестр Implementation Features переименован в OpenPGP Features Flags и его содержимое обновлено в соответствии с таблицей 11.

Реестр PGP User Attribute Types переименован в OpenPGP User Attribute Subpacket Types и его содержимое обновлено в соответствии с таблицей 13.

Реестр Image Format Subpacket Types переименован в OpenPGP Image Attribute Encoding Format и его содержимое обновлено в соответствии с таблицей 15.

Реестр Public Key Algorithms переименован в OpenPGP Public Key Algorithms и его содержимое обновлено в соответствии с таблицей 18.

Реестр Symmetric Key Algorithms переименован в OpenPGP Symmetric Key Algorithms и его содержимое обновлено в соответствии с таблицей 21.

Реестр Compression Algorithms переименован в OpenPGP Compression Algorithms и его содержимое обновлено в соответствии с таблицей 22.

Реестр Hash Algorithms переименован в OpenPGP Hash Algorithms и его содержимое обновлено в соответствии с таблицей 23.

15.3. Удалённый реестр

Агентство IANA пометило пустой реестр New Packet Versions как устаревший (OBSOLETE). В группу протоколов OpenPGP добавлено примечание:

> Тем, кто хотел использовать удалённый реестр New Packet Versions следует вместо этого регистрировать новые версии соответствующих пакетов в реестрах OpenPGP Key and Signature Versions, OpenPGP Key IDs and Fingerprints и OpenPGP Encrypted Message Packet Versions.

15.4. Добавленные реестры

Агентство IANA добавило в группу протоколов OpenPGP указанные ниже реестры. Исходное содержимое каждого из реестров указано в соответствующей таблице.

- OpenPGP Secret Key Encryption (S2K Usage Octet) (таблица 2).
- OpenPGP Signature Types (таблица 4).
- OpenPGP Signature Notation Data Subpacket Notation Flags (таблица 6).
- OpenPGP Signature Notation Data Subpacket Types (таблица 7).
- OpenPGP Key IDs and Fingerprints (таблица 12).
- OpenPGP Image Attribute Versions (таблица 14).
- OpenPGP Armor Header Lines (таблица 16).
- OpenPGP Armor Header Keys (таблица 17).
- OpenPGP ECC Curve OIDs and Usage (таблица 19).
- OpenPGP ECC Curve-Specific Wire Formats (таблица 20).
- OpenPGP Hash Algorithm Identifiers for RSA Signatures“ Use of EMSA-PKCS1-v1_5 Padding (таблица 24).
- OpenPGP AEAD Algorithms (таблица 25).
- OpenPGP Encrypted Message Packet Versions (таблица 26).
- OpenPGP Key and Signature Versions (таблица 27).
- OpenPGP Elliptic Curve Point Wire Formats (таблица 28).
- OpenPGP Elliptic Curve Scalar Encodings (таблица 29).
- OpenPGP ECDH KDF and KEK Parameters (таблица 30).

15.5. Правила регистрации

Во всех реестрах группы протоколов OpenPGP, за исключением указанных в параграфе 15.5.1, применяется процедура Specification Required, описанная в параграфе 4.6 [RFC8126]. Это подразумевает требование рецензирования и одобрения назначенными экспертами, а идентификаторы и их назначение должны быть подробно описаны в стабильной общедоступной спецификации, чтобы обеспечить возможность взаимодействия независимых реализаций.

15.5.1. Реестры с процедурой RFC Required

Ниже перечислены реестры, для которых применяется процедура RFC Required, описанная в параграфе 4.7 [RFC8126].

- OpenPGP Packet Types (таблица 3).
- OpenPGP Key IDs and Fingerprints (таблица 12).
- OpenPGP Encrypted Message Packet Versions (таблица 26).
- OpenPGP Key and Signature Versions (таблица 27).

15.6. Назначенные эксперты

Назначенные эксперты проверяют сохранение заново регистрируемыми элементами свойств безопасности, ожидаемых базовой реализацией, и отсутствие проблем совместимости с имеющимися реализациями сверх того, что они не создают или не воспринимают идентификаторы, связанные с регистрацией. Предложения, которые не соответствуют этим критериям, могут быть переданы в качестве новых заданий рабочей группе OpenPGP или её преемнику.

В последующих параграфах даны конкретные рекомендации для разных обновлений реестров, которые рассматривают назначенные эксперты. Таким экспертам при рассмотрении дополнений в группу протоколов OpenPGP также следует принимать во внимание параграф 12.11.

15.6.1. Версии ключей и подписей

При определении новых версий ключей или подписей OpenPGP следует обновлять реестр OpenPGP Key and Signature Versions (таблица 27). При задании новой версии OpenPGP Key следует также обновить реестр OpenPGP Key IDs and Fingerprints (таблица 12).

15.6.2. Версии шифрования

При определении новой версии пакета SEIPD (параграф 5.13), Public Key Encrypted Session Key (параграф 5.1) и/или Symmetric Key Encrypted Session Key (параграф 5.3) следует обновить реестр OpenPGP Encrypted Message Packet Versions (таблица 26). При обновлении SEIPD рассматривается также вопрос добавления соответствующего флага в реестр OpenPGP Features Flags (таблица 11).

15.6.3. Алгоритмы

В разделе указаны криптографические алгоритмы и алгоритмы сжатия, применяемые в OpenPGP. Добавление новых алгоритмов обычно просто и в некоторых случаях достаточно выделить идентификатор и указать ссылку на документ. При добавлении алгоритмов в некоторые реестры могут возникать дополнительные нюансы.

15.6.3.1. Алгоритмы с эллиптическими кривыми

При регистрации новой эллиптической кривой для OpenPGP необходимо зарегистрировать её OID в реестре OpenPGP ECC Curve OIDs and Usage (таблица 19), формат передачи - в реестре OpenPGP ECC Curve-Specific Wire Formats (таблица 20), а при использовании ECDH параметры KDF и KEK должны быть включены в реестр OpenPGP ECDH KDF and KEK Parameters (таблица 30). Если форматы передачи ещё не включены в реестр OpenPGP Elliptic Curve Point Wire Formats (таблица 28) или OpenPGP Elliptic Curve Scalar Encodings (таблица 29), это следует сделать.

15.6.3.2. Алгоритмы с симметричным ключом

При регистрации нового симметричного шифра с размером блока 64 или 128 битов и размером ключа, кратным 64 битам, новое рассмотрение не требуется. Если новый шифр имеет иной размер блока, требуется дополнительная документация, описывающая использование шифра в режиме CFB. Если в новом шифре ключ имеет необычный размер, требуется рассмотреть дополнение для упаковки ключей X25519 и X448, в которой в настоящее время дополнение не применяется.

15.6.3.3. Алгоритмы хэширования

Если регистрируемый в реестре OpenPGP Hash Algorithms (таблица 23) новый алгоритм хэширования применяется в схемах подписи RSA, он должен также иметь запись в реестре OpenPGP Hash Algorithm Identifiers for RSA Signatures“ Use of EMSA-PKCS1-v1_5 Padding (таблица 24).

16. Литература

16.1. Нормативные документы

[AES] NIST, «Advanced Encryption Standard (AES)», Updated May 2023, FIPS PUB 197, DOI 10.6028/NIST.FIPS.197-upd1, November 2001, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf>.

[BLOWFISH] Schneier, B., «Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)», Fast Software Encryption, Cambridge Security Workshop Proceedings, pp. 191-204, December 1993, <https://www.schneier.com/academic/archives/1994/09/description_of_a_new.html>.

[BZ2] bzip2, «bzip2 and libbzip2», 2010, <https://sourceware.org/bzip2/>.

[EAX] Bellare, M., Rogaway, P., and D. Wagner, «A Conventional Authenticated-Encryption Mode», April 2003, <https://seclab.cs.ucdavis.edu/papers/eax.pdf>.

[ELGAMAL] Elgamal, T., «A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms», IEEE Transactions on Information Theory, Vol. 31, Issue 4, pp. 469-472, DOI 10.1109/TIT.1985.1057074, July 1985, <https://doi.org/10.1109/TIT.1985.1057074>.

[FIPS180] NIST, «Secure Hash Standard (SHS)», FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.180-4.pdf>.

[FIPS186] NIST, «Digital Signature Standard (DSS)», FIPS PUB 186-5, DOI 10.6028/NIST.FIPS.186-5, February 2023, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf>.

[FIPS202] NIST, «SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions», FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, August 2015, <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf>.

[IDEA] Lai, X. And J. L. Massey, «A Proposal for a New Block Encryption Standard», Advances in Cryptology - EUROCRYPT „90, Vol. 473, pp. 389-404, DOI 10.1007/3-540-46877-3_35, January 1991, <https://link.springer.com/chapter/10.1007/3-540-46877-3_35>.

[ISO10646] ISO, «Information technology - Universal coded character set (UCS)», ISO/IEC 10646:2020, December 2020, <https://www.iso.org/standard/76835.html>.

[JFIF] ITU-T, «Information technology - Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF)», Recommendation ITU-T T.871, May 2011, <https://www.itu.int/rec/T-REC-T.871-201105-I>.

[RFC1321] Rivest, R., «The MD5 Message-Digest Algorithm», RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC1950] Deutsch, P. And J. Gailly, «ZLIB Compressed Data Format Specification version 3.3», RFC 1950, DOI 10.17487/RFC1950, May 1996, <https://www.rfc-editor.org/info/rfc1950>.

[RFC1951] Deutsch, P., «DEFLATE Compressed Data Format Specification version 1.3», RFC 1951, DOI 10.17487/RFC1951, May 1996, <https://www.rfc-editor.org/info/rfc1951>.

[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>.

[RFC2144] Adams, C., «The CAST-128 Encryption Algorithm», RFC 2144, DOI 10.17487/RFC2144, May 1997, <https://www.rfc-editor.org/info/rfc2144>.

[RFC2822] Resnick, P., Ed., «Internet Message Format», RFC 2822, DOI 10.17487/RFC2822, April 2001, <https://www.rfc-editor.org/info/rfc2822>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, «MIME Security with OpenPGP», RFC 3156, DOI 10.17487/RFC3156, August 2001, <https://www.rfc-editor.org/info/rfc3156>.

[RFC3394] Schaad, J. And R. Housley, «Advanced Encryption Standard (AES) Key Wrap Algorithm», RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/info/rfc3394>.

[RFC3629] Yergeau, F., «UTF-8, a transformation format of ISO 10646», STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3713] Matsui, M., Nakajima, J., and S. Moriai, «A Description of the Camellia Encryption Algorithm», RFC 3713, DOI 10.17487/RFC3713, April 2004, <https://www.rfc-editor.org/info/rfc3713>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, «Randomness Requirements for Security», BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4648] Josefsson, S., «The Base16, Base32, and Base64 Data Encodings», RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5322] Resnick, P., Ed., «Internet Message Format», RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC6234] Eastlake 3rd, D. And T. Hansen, «US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)», RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC7253] Krovetz, T. And P. Rogaway, «The OCB Authenticated-Encryption Algorithm», RFC 7253, DOI 10.17487/RFC7253, May 2014, <https://www.rfc-editor.org/info/rfc7253>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, «Elliptic Curves for Security», RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, «PKCS #1: RSA Cryptography Specifications Version 2.2», RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, «PKCS #5: Password-Based Cryptography Specification Version 2.1», RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[RFC8032] Josefsson, S. and I. Liusvaara, «Edwards-Curve Digital Signature Algorithm (EdDSA)», RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[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>.

[RFC9106] Biryukov, A., Dinu, D., Khovratovich, D., and S. Josefsson, «Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications», RFC 9106, DOI 10.17487/RFC9106, September 2021, <https://www.rfc-editor.org/info/rfc9106>.

[RIPEMD-160] ISO, «Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions», ISO/IEC 10118-3:1998, May 1998.

[SP800-38A] NIST, «Recommendation for Block Cipher Modes of Operation: Methods and Techniques», NIST Special Publication 800-38A, DOI 10.6028/NIST.SP.800-38A, December 2001, <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38a.pdf>.

[SP800-38D] NIST, «Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC», NIST Special Publication 800-38D, DOI 10.6028/NIST.SP.800-38D, November 2007, <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>.

[SP800-56A] NIST, «Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography», NIST Special Publication 800-56A Revision 3, DOI 10.6028/NIST.SP.800-56Ar, April 2018, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/nist.sp.800-56Ar3.pdf>.

[SP800-67] NIST, «Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher», NIST Special Publication 800-67 Revision 2, DOI 10.6028/NIST.SP.800-67r2, November 2017, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-67r2.pdf>.

[TWOFISH] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and N. Ferguson, «Twofish: A 128-Bit Block Cipher», June 1998, <https://www.schneier.com/wp-content/uploads/2016/02/paper-twofish-paper.pdf>.

16.2. Дополнительная литература

[BLEICHENBACHER] Bleichenbacher, D., «Generating ElGamal Signatures Without Knowing the Secret Key», EUROCRYPT“96: International Conference on the Theory and Applications of Cryptographic Techniques Proceedings, Vol. 1070, pp. 10-18, May 1996.

[BLEICHENBACHER-PKCS1] Bleichenbacher, D., «Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1», CRYPTO „98: International Cryptology Conference Proceedings, Vol. 1462, pp. 1-12, August 1998, <http://archiv.infsec.ethz.ch/education/fs08/secsem/Bleichenbacher98.pdf>.

[C99] ISO, «Information technology - Programming languages: C», ISO/IEC 9899:2018, June 2018, <https://www.iso.org/standard/74528.html>.

[CHECKOWAY] Checkoway, S., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, RP., Rescorla, E., and H. Shacham, «A Systematic Analysis of the Juniper Dual EC Incident», Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, DOI 10.1145/2976749.2978395, October 2016, <https://doi.org/10.1145/2976749.2978395>.

[EFAIL] Poddebniak, D., Dresen, C., Müller, J., Ising, F., Schinzel, S., Friedberger, S., Somorovsky, J., and J. Schwenk, «Efail: Breaking S/MIME and OpenPGP Email Encryption using Exfiltration Channels», Proceedings of the 27th USENIX Security Symposium, August 2018, <https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-poddebniak.pdf>.

[Errata-2199] RFC Errata, Erratum ID 2199, RFC 4880, <https://www.rfc-editor.org/errata/eid2199>.

[Errata-2200] RFC Errata, Erratum ID 2200, RFC 4880, <https://www.rfc-editor.org/errata/eid2200>.

[Errata-2206] RFC Errata, Erratum ID 2206, RFC 4880, <https://www.rfc-editor.org/errata/eid2206>.

[Errata-2208] RFC Errata, Erratum ID 2208, RFC 4880, <https://www.rfc-editor.org/errata/eid2208>.

[Errata-2214] RFC Errata, Erratum ID 2214, RFC 4880, <https://www.rfc-editor.org/errata/eid2214>.

[Errata-2216] RFC Errata, Erratum ID 2216, RFC 4880, <https://www.rfc-editor.org/errata/eid2216>.

[Errata-2219] RFC Errata, Erratum ID 2219, RFC 4880, <https://www.rfc-editor.org/errata/eid2219>.

[Errata-2222] RFC Errata, Erratum ID 2222, RFC 4880, <https://www.rfc-editor.org/errata/eid2222>.

[Errata-2226] RFC Errata, Erratum ID 2226, RFC 4880, <https://www.rfc-editor.org/errata/eid2226>.

[Errata-2234] RFC Errata, Erratum ID 2234, RFC 4880, <https://www.rfc-editor.org/errata/eid2234>.

[Errata-2235] RFC Errata, Erratum ID 2235, RFC 4880, <https://www.rfc-editor.org/errata/eid2235>.

[Errata-2236] RFC Errata, Erratum ID 2236, RFC 4880, <https://www.rfc-editor.org/errata/eid2236>.

[Errata-2238] RFC Errata, Erratum ID 2238, RFC 4880, <https://www.rfc-editor.org/errata/eid2238>.

[Errata-2240] RFC Errata, Erratum ID 2240, RFC 4880, <https://www.rfc-editor.org/errata/eid2240>.

[Errata-2242] RFC Errata, Erratum ID 2242, RFC 4880, <https://www.rfc-editor.org/errata/eid2242>.

[Errata-2243] RFC Errata, Erratum ID 2243, RFC 4880, <https://www.rfc-editor.org/errata/eid2243>.

[Errata-2270] RFC Errata, Erratum ID 2270, RFC 4880, <https://www.rfc-editor.org/errata/eid2270>.

[Errata-2271] RFC Errata, Erratum ID 2271, RFC 4880, <https://www.rfc-editor.org/errata/eid2271>.

[Errata-3298] RFC Errata, Erratum ID 3298, RFC 4880, <https://www.rfc-editor.org/errata/eid3298>.

[Errata-5491] RFC Errata, Erratum ID 5491, RFC 4880, <https://www.rfc-editor.org/errata/eid5491>.

[Errata-7545] RFC Errata, Erratum ID 7545, RFC 4880, <https://www.rfc-editor.org/errata/eid7545>.

[Errata-7889] RFC Errata, Erratum ID 7889, RFC 4880, <https://www.rfc-editor.org/errata/eid7889>.

[HASTAD] Hastad, J., «Solving Simultaneous Modular Equations of Low Degree», DOI 10.1137/0217019, April 1988, <https://doi.org/10.1137/0217019>.

[JKS02] Jallad, K., Katz, J., and B. Schneier, «Implementation of Chosen-Ciphertext Attacks against PGP and GnuPG», DOI 0.1007/3-540-45811-5_7, September 2002, <https://www.schneier.com/academic/archives/2002/01/implementation_of_ch.html>.

[KOBLITZ] Koblitz, N., «A course in number theory and cryptography», Chapter VI: Elliptic Curves, DOI 10.2307/3618498, 1997, <https://doi.org/10.2307/3618498>.

[KOPENPGP] Bruseghini, L., Paterson, K. G., and D. Huigens, «Victory by KO: Attacking OpenPGP Using Key Overwriting», Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 411-423, DOI 10.1145/3548606.3559363, November 2022, <https://dl.acm.org/doi/10.1145/3548606.3559363>.

[KR02] Klíma, V. And T. Rosa, «Attack on Private Signature Keys of the OpenPGP Format, PGP(TM) Programs and Other Applications Compatible with OpenPGP», Cryptology ePrint Archive, Paper 2002/076, March 2001, <https://eprint.iacr.org/2002/076>.

[MRLG15] Maury, F., Reinhard, JR., Levillain, O., and H. Gilbert, «Format Oracles on OpenPGP», Topics in Cryptology - CT- RSA 2015, Vol. 9048, pp. 220-236, DOI 10.1007/978-3-319-16715-2_12, January 2015, <https://doi.org/10.1007/978-3-319-16715-2_12>.

[MZ05] Mister, S. and R. Zuccherato, «An Attack on CFB Mode Encryption As Used By OpenPGP», Cryptology ePrint Archive, Paper 2005/033, February 2005, <http://eprint.iacr.org/2005/033>.

[OPENPGPCARD] Pietig, A., «Functional Specification of the OpenPGP application on ISO Smart Card Operating Systems», Version 3.4.1, March 2020, <https://gnupg.org/ftp/specs/OpenPGP-smart-card-application-3.4.1.pdf>.

[PAX] The Open Group, «The Open Group Base Specifications», „pax - portable archive interchange“, Issue 7, 2018 Edition, IEEE Std 1003.1-2017, 2018, <https://pubs.opengroup.org/onlinepubs/9699919799/utilities/pax.html>.

[PSSLR17] Poddebniak, D., Somorovsky, J., Schinzel, S., Lochter, M., and P. Rösler, «Attacking Deterministic Signature Schemes using Fault Attacks», Cryptology ePrint Archive, Paper 2017/1014, October 2017, <https://eprint.iacr.org/2017/1014>.

[REGEX] regex, «Henry Spencer“s regular expression libraries», <https://garyhouston.github.io/regex/>.

[RFC1991] Atkins, D., Stallings, W., and P. Zimmermann, «PGP Message Exchange Formats», RFC 1991, DOI 10.17487/RFC1991, August 1996, <https://www.rfc-editor.org/info/rfc1991>.

[RFC2440] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, «OpenPGP Message Format», RFC 2440, DOI 10.17487/RFC2440, November 1998, <https://www.rfc-editor.org/info/rfc2440>.

[RFC2978] Freed, N. And J. Postel, «IANA Charset Registration Procedures», BCP 19, RFC 2978, DOI 10.17487/RFC2978, October 2000, <https://www.rfc-editor.org/info/rfc2978>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, «OpenPGP Message Format», RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/info/rfc4880>.

[RFC5581] Shaw, D., «The Camellia Cipher in OpenPGP», RFC 5581, DOI 10.17487/RFC5581, June 2009, <https://www.rfc-editor.org/info/rfc5581>.

[RFC5639] Lochter, M. And J. Merkle, «Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation», RFC 5639, DOI 10.17487/RFC5639, March 2010, <https://www.rfc-editor.org/info/rfc5639>.

[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>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, «Fundamental Elliptic Curve Cryptography Algorithms», RFC 6090, DOI 10.17487/RFC6090, February 2011, <https://www.rfc-editor.org/info/rfc6090>.

[RFC6637] Jivsov, A., «Elliptic Curve Cryptography (ECC) in OpenPGP», RFC 6637, DOI 10.17487/RFC6637, June 2012, <https://www.rfc-editor.org/info/rfc6637>.

[SEC1] Standards for Efficient Cryptography Group, «SEC 1: Elliptic Curve Cryptography», May 2009, <https://www.secg.org/sec1-v2.pdf>.

[SHA1CD] »sha1collisiondetection», commit b4a7b0b, December 2020, <https://github.com/cr-marcstevens/sha1collisiondetection>.

[SHAMBLES] Leurent, G. And T. Peyrin, «Sha-1 is a shambles: first chosen-prefix collision on sha-1 and application to the PGP web of trust», August 2020, <https://dl.acm.org/doi/abs/10.5555/3489212.3489316/>.

[SP800-131A] NIST, «Transitioning the Use of Cryptographic Algorithms and Key Lengths», NIST Special Publication 800-131A, Revision 2, DOI 10.6028/NIST.SP.800-131Ar2, March 2019, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf>.

[SP800-57] NIST, «Recommendation for Key Management: Part 1 - General», NIST Special Publication 800-57 Part 1, Revision 5, DOI 10.6028/NIST.SP.800-57pt1r5, May 2020, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>.

[STEVENS2013] Stevens, M., «Counter-cryptanalysis», Cryptology ePrint Archive, Paper 2013/358, June 2013, <https://eprint.iacr.org/2013/358>.

[UNIFIED-DIFF] Free Software Foundation, «Comparing and Merging Files», „Detailed Description of Unified Format“, Section 2.2.2.2, January 2021, <https://www.gnu.org/software/diffutils/manual/html_node/Detailed-Unified.html>.

[USENIX-STUDY] Dechand, S., Schürmann, D., Busse, K., Acar, Y., Fahl, S., and M. Smith, «An Empirical Study of Textual Key-Fingerprint Representations», ISBN 978-1-931971-32-4, August 2016, <https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_dechand.pdf>.

Приложение A. Тестовые векторы

Ниже приведён ряд ненормативных примеров для оказания помощи в реализации этой спецификации.

A.1. Пример ключа Ed25519Legacy версии 4

Секретный ключ имеет вид

```
D: 1a8b1ff05ded48e18bf50166c664ab023ea70003d78d9e41f5758a91d850f8d2
```

Отметим, что это необработанный (raw) секретный ключ, подаваемый на вход операции подписи EdDSA. Ключ был создан 19 августа 2014 г в 14:28:27 и оттиск ключа OpenPGP имеет вид

```
    C959 BDBA FA32 A2F8 9A15  3B67 8CFD E121 9796 5A9A
```

Связанные с алгоритмом входные параметры без заголовков размера MPI имеют вид

```
oid: 2b06010401da470f01
q: 403f098994bdd916ed4053197934e4a87c80733a1280d62f8010992e43ee3b2406
```

Весь пакет Public Key будет иметь вид

```
    98 33 04 53 f3 5f 0b 16  09 2b 06 01 04 01 da 47
    0f 01 01 07 40 3f 09 89  94 bd d9 16 ed 40 53 19
    79 34 e4 a8 7c 80 73 3a  12 80 d6 2f 80 10 99 2e
    43 ee 3b 24 06
```

А в формате ASCII-armored он представляется как

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

xjMEU/NfCxYJKwYBBAHaRw8BAQdAPwmJlL3ZFu1AUxl5NOSofIBzOhKA1i+AEJkuQ+47JAY=
-----END PGP PUBLIC KEY BLOCK-----
```

A.2. Пример подписи Ed25519Legacy версии 4

Подпись создаётся с использованием примера ключа для входных данных OpenPGP от 16 сентября 2015 г. 12:24:53 UTC, поэтому ввод хэш-функции имеет вид

```
m: 4f70656e5047500400160800060502 55f95f9504ff0000000c
```

Использование алгоритма хэширования SHA2-256 даёт дайджест

```
d: f6220a3f757814f4c2176ffbb68b00249cd4ccdc059c4b34ad871f30b1740280
```

который передаётся функции подписи EdDSA, дающей на выходе

```
r: 56f90cca98e2102637bd983fdb16c131dfd27ed82bf4dde5606e0d756aed3366
s: d09c4fa11527f038e0f57f2201d82f2ea2c9033265fa6ceb489e854bae61b404
```

Пакет Signature целиком имеет вид

```
    88 5e 04 00 16 08 00 06  05 02 55 f9 5f 95 00 0a
    09 10 8c fd e1 21 97 96  5a 9a f6 22 00 ff 56 f9
    0c ca 98 e2 10 26 37 bd  98 3f db 16 c1 31 df d2
    7e d8 2b f4 dd e5 60 6e  0d 75 6a ed 33 66 01 00
    d0 9c 4f a1 15 27 f0 38  e0 f5 7f 22 01 d8 2f 2e
    a2 c9 03 32 65 fa 6c eb  48 9e 85 4b ae 61 b4 04
```

В форме ASCII-armored этот пакет имеет фид

```
-----BEGIN PGP SIGNATURE-----

iF4EABYIAAYFAlX5X5UACgkQjP3hIZeWWpr2IgD/VvkMypjiECY3vZg/2xbBMd/S
ftgr9N3lYG4NdWrtM2YBANCcT6EVJ/A44PV/IgHYLy6iyQMyZfps60iehUuuYbQE
-----END PGP SIGNATURE-----
```

A.3. Пример сертификата версии 6 (переносимый открытый ключ)

Переносимый открытый ключ содержит:

- пакет открытого ключа Ed25519 версии 6
- самоподпись Direct Key версии 6
- пакет открытого субключа X25519 версии 6
- подпись Subkey Binding версии 6

Первичный ключ имеет оттиск

```
CB186C4F0609A697E4D52DFA6C722B0C1F1E27C18A56708F6525EC27BAD9ACC9
```

Субключ имеет оттиск

```
12C83F1E706F6308FE151A417743A1F033790E93E9978488D1DB378DA9930885
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xioGY4d/4xsAAAAg+U2nu0jWCmHlZ3BqZYfQMxmZu52JGggkLq2EVD34laPCsQYf
GwoAAABCBYJjh3/jAwsJBwUVCg4IDAIWAAKbAwIeCSIhBssYbE8GCaaX5NUt+mxy
KwwfHifBilZwj2Ul7Ce62azJBScJAgcCAAAAAK0oIBA+LX0ifsDm185Ecds2v8lw
gyU2kCcUmKfvBXbAf6rhRYWzuQOwEn7E/aLwIwRaLsdry0+VcallHhSu4RN6HWaE
QsiPlR4zxP/TP7mhfVEe7XWPxtnMUMtf15OyA51YBM4qBmOHf+MZAAAAIIaTJINn
+eUBXbki+PSAld2nhJh/LvmFsS+60WyvXkQ1wpsGGBsKAAAALAWCY4d/4wKbDCIh
BssYbE8GCaaX5NUt+mxyKwwfHifBilZwj2Ul7Ce62azJAAAAAAQBIKbpGG2dWTX8
j+VjFM21J0hqWlEg+bdiojWnKfA5AQpWUWtnNwDEM0g12vYxoWM8Y81W+bHBw805
I8kWVkXU6vFOi+Hwvv/ira7ofJu16NnoUkhclkUrk0mXubZvyl4GBg==
-----END PGP PUBLIC KEY BLOCK-----
```

Соответствующий переносимый секретный ключ представлен в Приложении A.4.

A.3.1. Поток хэшированных данных для проверки подписи

Самоподпись Direct Key в сертификате из Приложения A.3 выполняется для последовательности данных

```
0x0000  10 3e 2d 7d 22 7e c0 e6
0x0008  d7 ce 44 71 db 36 bf c9
0x0010  70 83 25 36 90 27 14 98
0x0018  a7 ef 05 76 c0 7f aa e1
0x0020  9b 00 00 00 2a 06 63 87
0x0028  7f e3 1b 00 00 00 20 f9
0x0030  4d a7 bb 48 d6 0a 61 e5
0x0038  67 70 6a 65 87 d0 33 19
0x0040  99 bb 9d 89 1a 08 24 2e
0x0048  ad 84 54 3d f8 95 a3 06
0x0050  1f 1b 0a 00 00 00 42 05
0x0058  82 63 87 7f e3 03 0b 09
0x0060  07 05 15 0a 0e 08 0c 02
0x0068  16 00 02 9b 03 02 1e 09
0x0070  22 21 06 cb 18 6c 4f 06
0x0078  09 a6 97 e4 d5 2d fa 6c
0x0080  72 2b 0c 1f 1e 27 c1 8a
0x0088  56 70 8f 65 25 ec 27 ba
0x0090  d9 ac c9 05 27 09 02 07
0x0098  02 06 ff 00 00 00 4a
```

Те же данные с разбиением по октетам и описанием семантики представлены ниже.

```
0x0000  10 3e 2d 7d 22 7e c0 e6  затравка (salt)
0x0008  d7 ce 44 71 db 36 bf c9
0x0010  70 83 25 36 90 27 14 98
0x0018  a7 ef 05 76 c0 7f aa e1
        [ начало открытого ключа ]
0x0020  9b                       пакет ключа
0x0021     00 00 00 2a           размер открытого ключа
0x0025                 06        версия открытого ключа
0x0026                    63 87  время создания
0x0028  7f e3                      (2022-11-30T16:08:03Z)
0x002a        1b                 алгоритм ключа Ed25519
0x002b           00 00 00 20     размер ключа
0x002f                       f9  открытый ключ Ed25519
0x0030  4d a7 bb 48 d6 0a 61 e5
0x0038  67 70 6a 65 87 d0 33 19
0x0040  99 bb 9d 89 1a 08 24 2e
0x0048  ad 84 54 3d f8 95 a3
         [ начало трейлера ]
0x004f                       06  подпись версии 6
0x0050  1f                       тип подписи Direct Key
0x0051     1b                    алгоритм подписи Ed25519
0x0052        0a                 алгоритм хэширования SHA2-512
0x0053           00 00 00 42     размер хэшированных субпакетов
0x0057                       05  размер субпакета
0x0058  82                       критичный субпакет Sig Creation Time
0x0059     63 87 7f e3           время создания подписи
0x005d                 03        размер субпакета
0x005e                    0b     тип субпакета Pref. V1 SEIPD Ciphers
0x005f                       09  Ciphers: [AES256 AES128]
0x0060  07
0x0061     05                    размер субпакета
0x0062        15                 тип субпакета Pref. Hash Algorithms
0x0063           0a 0e           хеши [SHA2-512 SHA3-512
0x0065                 08 0c               SHA2-256 SHA3-256]
0x0067                       02  размер субпакета
0x0068  16                       тип субпакета Pref. Compression
0x0069     00                    сжатие [none]
0x006a        02                 размер субпакета
0x006b           9b              критичный субпакет Key Flags
0x006c              03           флаги ключа {certify, sign}
0x006d                 02        размер субпакета
0x006e                    1e     тип субпакета Features
0x006f                       09  свойства {v1SEIPD, v2SEIPD}
0x0070  22                       размер субпакета
0x0071     21                    тип субпакета Issuer Fingerprint
0x0072        06                 оттиск версии 6
0x0073           cb 18 6c 4f 06  оттиск
0x0078  09 a6 97 e4 d5 2d fa 6c
```

```
0x0080  72 2b 0c 1f 1e 27 c1 8a
0x0088  56 70 8f 65 25 ec 27 ba
0x0090  d9 ac c9
0x0093           05              размер субпакета
0x0094              27           тип субпакета Pref. AEAD Ciphersuites
0x0095                 09 02 07  шифры
0x0098  02                         [ AES256-OCB, AES128-OCB ]
0x0099     06                     подпись версии 6
0x009a        ff                  сторожевой октет (sentinel)
0x009b           00 00 00 4a      размер трейлера
```

Подпись привязки субключа (Subkey Binding) в Приложении A.3 создана для указанной ниже последовательности данных.

```
0x0000  a6 e9 18 6d 9d 59 35 fc
0x0008  8f e5 63 14 cd b5 27 48
0x0010  6a 5a 51 20 f9 b7 62 a2
0x0018  35 a7 29 f0 39 01 0a 56
0x0020  9b 00 00 00 2a 06 63 87
0x0028  7f e3 1b 00 00 00 20 f9
0x0030  4d a7 bb 48 d6 0a 61 e5
0x0038  67 70 6a 65 87 d0 33 19
0x0040  99 bb 9d 89 1a 08 24 2e
0x0048  ad 84 54 3d f8 95 a3 9b
0x0050  00 00 00 2a 06 63 87 7f
0x0058  e3 19 00 00 00 20 86 93
0x0060  24 83 67 f9 e5 01 5d b9
0x0068  22 f8 f4 80 95 dd a7 84
0x0070  98 7f 2d 59 85 b1 2f ba
0x0078  d1 6c af 5e 44 35 06 18
0x0080  1b 0a 00 00 00 2c 05 82
0x0088  63 87 7f e3 02 9b 0c 22
0x0090  21 06 cb 18 6c 4f 06 09
0x0098  a6 97 e4 d5 2d fa 6c 72
0x00a0  2b 0c 1f 1e 27 c1 8a 56
0x00a8  70 8f 65 25 ec 27 ba d9
0x00b0  ac c9 06 ff 00 00 00 34
```

Эти данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  a6 e9 18 6d 9d 59 35 fc  затравка (salt)
0x0008  8f e5 63 14 cd b5 27 48
0x0010  6a 5a 51 20 f9 b7 62 a2
0x0018  35 a7 29 f0 39 01 0a 56
      [ начало первичного открытого ключа ]
0x0020  9b                       пакет ключа
0x0021     00 00 00 2a           размер открытого ключа
0x0025                 06        версия открытого ключа
0x0026                    63 87  время создания ключа
0x0028  7f e3                      (2022-11-30T16:08:03Z)
0x002a        1b                 алгоритм ключа Ed25519
0x002b           00 00 00 20     размер ключа
0x002f                       f9  открытый ключ Ed25519
0x0030  4d a7 bb 48 d6 0a 61 e5
0x0038  67 70 6a 65 87 d0 33 19
0x0040  99 bb 9d 89 1a 08 24 2e
0x0048  ad 84 54 3d f8 95 a3
      [ начало субключа ]
0x004f                       9b  пакет ключа
0x0050  00 00 00 2a              размер открытого ключа
0x0054              06           версия открытого ключа
0x0055                 63 87 7f  время создания (2022-11-30T16:08:03Z)
0x0058  e3
0x0059     19                    алгоритм ключа X25519
0x005a        00 00 00 20        размер ключа
0x005e                    86 93  открытый ключ X25519
0x0060  24 83 67 f9 e5 01 5d b9
0x0068  22 f8 f4 80 95 dd a7 84
0x0070  98 7f 2d 59 85 b1 2f ba
0x0078  d1 6c af 5e 44 35
       [ начало трейлера ]
0x007e                    06     подпись версии 6
0x007f                       18  тип подписи Subkey Binding
0x0080  1b                       алгоритм подписи Ed25519
0x0081     0a                    алгоритм хэширования SHA2-512
0x0082        00 00 00 2c        размер хэшированных субпакетов
0x0086                    05     размер субпакета
0x0087                       82  критичный субпакет Sig Creation Time
0x0088  63 87 7f e3              время создания подписи
0x008c              02           размер субпакета
0x008d                 9b        критичный субпакет Key Flags
0x008e                    0c     флаги ключа {EncComms, EncStorage}
0x008f                       22  размер субпакета
0x0090  21                       тип субпакета Issuer Fingerprint
0x0091     06                    оттиск версии 6
0x0092        cb 18 6c 4f 06 09  оттиск
0x0098  a6 97 e4 d5 2d fa 6c 72
0x00a0  2b 0c 1f 1e 27 c1 8a 56
```

```
0x00a8  70 8f 65 25 ec 27 ba d9
0x00b0  ac c9
0x00b2        06                подпись версии 6
0x00b3           ff             охранный октет (sentinel)
0x00b4              00 00 00 34  размер трейлера
```

A.4. Пример секретного ключа версии 6 (Transferable Secret Key)

Переносимый секретный ключ содержит:

- пакет секретного ключа Ed25519 версии 6
- самоподпись Direct Key версии 6
- пакет секретного субключа X25519 версии 6
- подпись привязки ключа версии 6

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

xUsGY4d/4xsAAAAg+U2nu0jWCmHlZ3BqZYfQMxmZu52JGggkLq2EVD34laMAGXKB
exK+cH6NX1hs5hNhIB00TrJmosgv3mg1ditlsLfCsQYfGwoAAABCBYJjh3/jAwsJ
BwUVCg4IDAIWAAKbAwIeCSIhBssYbE8GCaaX5NUt+mxyKwwfHifBilZwj2Ul7Ce6
2azJBScJAgcCAAAAAK0oIBA+LX0ifsDm185Ecds2v8lwgyU2kCcUmKfvBXbAf6rh
RYWzuQOwEn7E/aLwIwRaLsdry0+VcallHhSu4RN6HWaEQsiPlR4zxP/TP7mhfVEe
7XWPxtnMUMtf15OyA51YBMdLBmOHf+MZAAAAIIaTJINn+eUBXbki+PSAld2nhJh/
LvmFsS+60WyvXkQ1AE1gCk95TUR3XFeibg/u/tVY6a//1q0NWC1X+yui3O24wpsG
GBsKAAAALAWCY4d/4wKbDCIhBssYbE8GCaaX5NUt+mxyKwwfHifBilZwj2Ul7Ce6
2azJAAAAAAQBIKbpGG2dWTX8j+VjFM21J0hqWlEg+bdiojWnKfA5AQpWUWtnNwDE
M0g12vYxoWM8Y81W+bHBw805I8kWVkXU6vFOi+Hwvv/ira7ofJu16NnoUkhclkUr
k0mXubZvyl4GBg==
-----END PGP PRIVATE KEY BLOCK-----
```

Соответствующий переносимый открытый ключ (Transferable Public Key) представлен в Приложении A.3.

A.5. Пример блокированного секретного ключа версии 6 (переносимого)

Здесь применяется секретный ключ из Приложения A.4, но секретный ключевой материал защищён парольной фразой с использованием AEAD и Argon2. Парольной фразой является строка ASCII «correct horse battery staple»

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

xYIGY4d/4xsAAAAg+U2nu0jWCmHlZ3BqZYfQMxmZu52JGggkLq2EVD34laP9JgkC
FARdb9ccngltHraRe25uHuyuAQQVtKipJ0+r5jL4dacGWSAheCWPpITYiyfyIOPS
3gIDyg8f7strd1OB4+LzsUhcIjOMpVHgmiY/IutJkulneoBYwrEGHxsKAAAAQgWC
Y4d/4wMLCQcFFQoOCAwCFgACmwMCHgkiIQbLGGxPBgmml+TVLfpscisMHx4nwYpW
cI9lJewnutmsyQUnCQIHAgAAAACtKCAQPi19In7A5tfORHHbNr/JcIMlNpAnFJin
7wV2wH+q4UWFs7kDsBJ+xP2i8CMEWi7Ha8tPlXGpZR4UruETeh1mhELIj5UeM8T/
0z+5oX1RHu11j8bZzFDLX9eTsgOdWATHggZjh3/jGQAAACCGkySDZ/nlAV25Ivj0
gJXdp4SYfy1ZhbEvutFsr15ENf0mCQIUBA5hhGgp2oaavg6mFUXcFMwBBBUuE8qf
9Ock+xwusd+GAglBr5LVyr/lup3xxQvHXFSjjA2haXfoN6xUGRdDEHI6+uevKjVR
v5oAxgu7eJpaXNjCmwYYGwoAAAAsBYJjh3/jApsMIiEGyxhsTwYJppfk1S36bHIr
DB8eJ8GKVnCPZSXsJ7rZrMkAAAAABAEgpukYbZ1ZNfyP5WMUzbUnSGpaUSD5t2Ki
Nacp8DkBClZRa2c3AMQzSDXa9jGhYzxjzVb5scHDzTkjyRZWRdTq8U6L4da+/+Kt
ruh8m7Xo2ehSSFyWRSuTSZe5tm/KxgYG
-----END PGP PRIVATE KEY BLOCK-----
```

A.5.1. Промежуточные данные для блокированного первичного ключа

Полученный с помощью S2K материал для секретного ключа имеет вид

```
832bd2662a5c2b251ee3fc82aec349a766ca539015880133002e5a21960b3bcf
```

После HKDF симметричный ключевой материал для AEAD-шифрования секретного ключа имеет вид

```
9e37cb26787f37e18db172795c4c297550d39ac82511d9af4c8706db6a77fd51
```

Дополнительные данные AEAD для первичного ключа имеют вид

```
c50663877fe31b00000020f94da7bb48d60a61e567706a6587d0331999bb9d89
1a08242ead84543df895a3
```

A.5.2. Промежуточные данные для блокированного субключа

Полученный с помощью S2K материал для субключа имеет вид

```
f74a6ce873a089ef13a3da9ac059777bb22340d15eaa6c9dc0f8ef09035c67cd
```

После HKDF симметричный ключевой материал для AEAD-шифрования субключа имеет вид

```
3c60cb63285f62f4c3de49835786f011cf6f4c069f61232cd7013ff5fd31e603
```

Дополнительные данные AEAD для субключа имеют вид

```
c70663877fe3190000002086932483 67f9e5015db922f8f48095dda784987f2d
5985b12fbad16caf5e4435
```

A.6. Пример открытого сообщения с подписью

Ниже приведено сообщение, подписанное с помощью схемы подписи с открытым текстом (Cleartext Signature Framework, раздел 7). Его можно проверить с помощью сертификата из Приложения A.3.

Отметим, что в сообщении применяется экранирование дефисом (dash-escaping,параграф 7.2).

```
-----BEGIN PGP SIGNED MESSAGE-----

What we need from the grocery store:

- - tofu
- - vegetables
```

```
- - noodles

-----BEGIN PGP SIGNATURE-----

wpgGARsKAAAAKQWCY5ijYyIhBssYbE8GCaaX5NUt+mxyKwwfHifBilZwj2Ul7Ce6
2azJAAAAAGk2IHZJX1AhiJD39eLuPBgiUU9wUA9VHYblySHkBONKU/usJ9BvuAqo
/FvLFuGWMbKAdA+epq7V4HOtAPlBWmU8QOd6aud+aSunHQaaEJ+iTFjP2OMW0KBr
NK2ay45cX1IVAQ==
-----END PGP SIGNATURE-----
```

Пакет Signature представлен ниже

```
0x0000  c2                      тип пакета Signature
0x0001     98                   размер пакета
0x0002        06                подпись версии 6
0x0003           01             тип подписи Canonical Text
0x0004              1b          алгоритм открытого ключа Ed25519
0x0005                 0a       используемый алгоритм хэширования SHA2-512
0x0006                    00 00 размер хэшированных субпакетов 41
0x0008  00 29
0x000a        05                размер субпакета
0x000b           82             критичный субпакет Sig Creation Time
0x000c              63 98 a3 63  (2022-12-13T16:08:03Z)
0x0010  22                      размер субпакета
0x0011     21                   тип субпакета Issuer Fingerprint
0x0012        06                оттиск версии 6
0x0013           cb 18 6c 4f 06 Fingerprint
0x001a  09 a6 97 e4 d5 2d fa 6c
0x0020  72 2b 0c 1f 1e 27 c1 8a
0x0028  56 70 8f 65 25 ec 27 ba
0x0030  d9 ac c9
0x0033           00 00 00 00    размер нехэшированных субпакетов 0
0x0037                       69 оставшиеся 16 битов подписанного хэша
0x0038  36
0x0039     20                   размер затравки
0x003a        76 49 5f 50 21 88 затравка (salt)
0x0040  90 f7 f5 e2 ee 3c 18 22
0x0048  51 4f 70 50 0f 55 1d 86
0x0050  e5 c9 21 e4 04 e3 4a 53
0x0058  fb ac
0x005a        27 d0 6f b8 0a a8 подпись Ed25519
0x0060  fc 5b cb 16 e1 96 31 b2
0x0068  80 74 0f 9e a6 ae d5 e0
0x0070  73 ad 00 f9 41 5a 65 3c
0x0078  40 e7 7a 6a e7 7e 69 2b
0x0080  a7 1d 06 9a 10 9f a2 4c
0x0088  58 cf d8 e3 16 d0 a0 6b
0x0090  34 ad 9a cb 8e 5c 5f 52
0x0098  15 01
```

Подпись создаётся для приведённых ниже данных.

```
0x0000  76 49 5f 50 21 88 90 f7
0x0008  f5 e2 ee 3c 18 22 51 4f
0x0010  70 50 0f 55 1d 86 e5 c9
0x0018  21 e4 04 e3 4a 53 fb ac
0x0020  57 68 61 74 20 77 65 20
0x0028  6e 65 65 64 20 66 72 6f
0x0030  6d 20 74 68 65 20 67 72
0x0038  6f 63 65 72 79 20 73 74
0x0040  6f 72 65 3a 0d 0a 0d 0a
0x0048  2d 20 74 6f 66 75 0d 0a
0x0050  2d 20 76 65 67 65 74 61
0x0058  62 6c 65 73 0d 0a 2d 20
0x0060  6e 6f 6f 64 6c 65 73 0d
0x0068  0a 06 01 1b 0a 00 00 00
0x0070  29 05 82 63 98 a3 63 22
0x0078  21 06 cb 18 6c 4f 06 09
0x0080  a6 97 e4 d5 2d fa 6c 72
0x0088  2b 0c 1f 1e 27 c1 8a 56
0x0090  70 8f 65 25 ec 27 ba d9
0x0098  ac c9 06 ff 00 00 00 31
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  76 49 5f 50 21 88 90 f7  затравка (salt)
0x0008  f5 e2 ee 3c 18 22 51 4f
0x0010  70 50 0f 55 1d 86 e5 c9
0x0018  21 e4 04 e3 4a 53 fb ac
      [ начало сообщения ]
0x0020  57 68 61 74 20 77 65 20  канонизированное сообщение
0x0028  6e 65 65 64 20 66 72 6f
0x0030  6d 20 74 68 65 20 67 72
0x0038  6f 63 65 72 79 20 73 74
0x0040  6f 72 65 3a 0d 0a 0d 0a
0x0048  2d 20 74 6f 66 75 0d 0a
0x0050  2d 20 76 65 67 65 74 61
0x0058  62 6c 65 73 0d 0a 2d 20
0x0060  6e 6f 6f 64 6c 65 73 0d
```

```
0x0068  0a
      [ начало трейлера ]
0x0069     06                   подпись версии 6
0x006a        01                тип подписи Canonical Text
0x006b           1b             алгоритм открытого ключа  Ed25519
0x006c              0a          алгоритм хэширования SHA2-512
0x006d                 00 00 00 размер хэшированных субпакетов
0x0070  29
0x0071     05                   размер субпакета
0x0072        82                критичный субпакет Sig Creation Time
0x0073           63 98 a3 63       (2022-12-13T16:08:03Z)
0x0077                       22  размер субпакета
0x0078  21                       тип субпакета Issuer Fingerprint
0x0079     06                    оттиск версии 6
0x007a        cb 18 6c 4f 06 09  оттиск
0x0080  a6 97 e4 d5 2d fa 6c 72
0x0088  2b 0c 1f 1e 27 c1 8a 56
0x0090  70 8f 65 25 ec 27 ba d9
0x0098  ac c9
0x009a        06                подпись версии 6
0x009b           ff             сторожевой октет (sentinel)
0x009c              00 00 00 31 размер трейлера
```

Дайджест SHA2-512 для этих данных имеет вид

```
69365bf44a97af1f0844f1f6ab83fdf6b36f26692efaa621a8aac91c4e29ea07
e894cabc6e2f20eedfce6c03b89141a2cc7cbe245e6e7a5654addbec5000b89b
```

A.7. Пример сообщения со встроенной подписью

Ниже приведены сообщение и подпись как в Приложении A.6, но со встроенной подписью (inline-signed). Хэшированные данные, все промежуточные значения и шестнадцатеричные дампы совпадают.

```
-----BEGIN PGP MESSAGE-----

xEYGAQobIHZJX1AhiJD39eLuPBgiUU9wUA9VHYblySHkBONKU/usyxhsTwYJppfk
1S36bHIrDB8eJ8GKVnCPZSXsJ7rZrMkBy0p1AAAAAABXaGF0IHdlIG5lZWQgZnJv
bSB0aGUgZ3JvY2VyeSBzdG9yZToKCi0gdG9mdQotIHZlZ2V0YWJsZXMKLSBub29k
bGVzCsKYBgEbCgAAACkFgmOYo2MiIQbLGGxPBgmml+TVLfpscisMHx4nwYpWcI9l
JewnutmsyQAAAABpNiB2SV9QIYiQ9/Xi7jwYIlFPcFAPVR2G5ckh5ATjSlP7rCfQ
b7gKqPxbyxbhljGygHQPnqau1eBzrQD5QVplPEDnemrnfmkrpx0GmhCfokxYz9jj
FtCgazStmsuOXF9SFQE=
-----END PGP MESSAGE-----
```

A.8. Пример шифрования и расшифровки X25519-AEAD-OCB

В этом примере шифруется строка «Hello, world!» с сертификатом из Приложения A.3, используя AES-128 с AEAD-OCB.

A.8.1. Пример пакета зашифрованного с открытым ключом сеансового ключа v6

Октетное представление пакета приведено ниже.

```
0x0000  c1 5d 06 21 06 12 c8 3f
0x0008  1e 70 6f 63 08 fe 15 1a
0x0010  41 77 43 a1 f0 33 79 0e
0x0018  93 e9 97 84 88 d1 db 37
0x0020  8d a9 93 08 85 19 87 cf
0x0028  18 d5 f1 b5 3f 81 7c ce
0x0030  5a 00 4c f3 93 cc 89 58
0x0038  bd dc 06 5f 25 f8 4a f5
0x0040  09 b1 7d d3 67 64 18 de
0x0048  a3 55 43 79 56 61 79 01
0x0050  e0 69 57 fb ca 8a 6a 47
0x0058  a5 b5 15 3e 8d 3a b7
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  c1                       тип пакета PKESK
0x0001     5d                    размер пакета
0x0002        06                 v6 PKESK
0x0003           21              размер оттиска
0x0004              06           ключ версии 6
0x0005                 12 c8 3f  оттиск ключа
0x0008  1e 70 6f 63 08 fe 15 1a
0x0010  41 77 43 a1 f0 33 79 0e
0x0018  93 e9 97 84 88 d1 db 37
0x0020  8d a9 93 08 85
0x0025                 19        алгоритм X25519
0x0026                    87 cf  эфемерный ключ
0x0028  18 d5 f1 b5 3f 81 7c ce
0x0030  5a 00 4c f3 93 cc 89 58
0x0038  bd dc 06 5f 25 f8 4a f5
0x0040  09 b1 7d d3 67 64
0x0046                    18     размер ESK
0x0047                       de  ESK
0x0048  a3 55 43 79 56 61 79 01
0x0050  e0 69 57 fb ca 8a 6a 47
0x0058  a5 b5 15 3e 8d 3a b7
```

A.8.2. Шифрование и расшифровка X25519 для сеансового ключа

Эфемерный ключ

```
87 cf 18 d5 f1 b5 3f 81 7c ce 5a 00 4c f3 93 cc
89 58 bd dc 06 5f 25 f8 4a f5 09 b1 7d d3 67 64
```

Этот ключ выведен из представленного ниже эфемерного ключевого материала, который не передаётся в линию.

```
af 1e 43 c0 d1 23 ef e8 93 a7 d4 d3 90 f3 a7 61
e3 fa c3 3d fc 7f 3e da a8 30 c9 01 13 52 c7 79
```

Открытый ключ из целевого сертификата (см. Приложение A.3) имеет вид

```
86 93 24 83 67 f9 e5 01 5d b9 22 f8 f4 80 95 dd
a7 84 98 7f 2d 59 85 b1 2f ba d1 6c af 5e 44 35
```

Соответствующий долгосрочный секретный ключевой материал X25519 (см. Приложение A.4) имеет вид

```
4d 60 0a 4f 79 4d 44 77 5c 57 a2 6e 0f ee fe d5
58 e9 af ff d6 ad 0d 58 2d 57 fb 2b a2 dc ed b8
```

Общая точка

```
67 e3 0e 69 cd c7 ba b2 a2 68 0d 78 ac a4 6a 2f
8b 6e 2a e4 4d 39 8b dc 6f 92 c5 ad 4a 49 25 14
```

Выход HKDF

```
f6 6d ad cf f6 45 92 23 9b 25 45 39 b6 4f f6 07
```

Расшифрованный сеансовый ключ

```
dd 70 8f 6f a1 ed 65 11 4d 68 d2 34 3e 7c 2f 1d
```

A.8.3. Пример пакета SEIPD v2

Ниже представлена последовательность октетов пакета.

```
0x0000  d2 69 02 07 02 06 61 64
0x0008  16 53 5b e0 b0 71 6d 60
0x0010  e0 52 a5 6c 4c 40 7f 9e
0x0018  b3 6b 0e fa fe 9a d0 a0
0x0020  df 9b 03 3c 69 a2 1b a9
0x0028  eb d2 c0 ec 95 bf 56 9d
0x0030  25 c9 99 ee 4a 3d e1 70
0x0038  58 f4 0d fa 8b 4c 68 2b
0x0040  e3 fb bb d7 b2 7e b0 f5
0x0048  9b b5 00 5f 80 c7 c6 f4
0x0050  03 88 c3 0a d4 06 ab 05
0x0058  13 dc d6 f9 fd 73 76 56
0x0060  28 6e 11 77 d0 0f 88 8a
0x0068  db 31 c4
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  d2                      тип пакета SEIPD
0x0001     69                   размер пакета
0x0002        02                v2 SEIPD
0x0003           07             шифр AES128
0x0004              02          режим AEAD OCB
0x0005                 06       размер chunk (2^12 октетов)
0x0006                    61 64 затравка (salt)
0x0008  16 53 5b e0 b0 71 6d 60
0x0010  e0 52 a5 6c 4c 40 7f 9e
0x0018  b3 6b 0e fa fe 9a d0 a0
0x0020  df 9b 03 3c 69 a2
0x0026                    1b a9 chunk #0 шифрованные данные
0x0028  eb d2 c0 ec 95 bf 56 9d
0x0030  25 c9 99 ee 4a 3d e1 70
0x0038  58 f4 0d fa 8b 4c 68 2b
0x0040  e3 fb bb d7 b2 7e b0 f5
0x0048  9b b5 00
0x004b           5f 80 c7 c6 f4 chunk #0 тег AEAD
0x0050  03 88 c3 0a d4 06 ab 05
0x0058  13 dc d6
0x005b           f9 fd 73 76 56 финальный тег AEAD (#1)
S0x0060  28 6e 11 77 d0 0f 88 8a
0x0068  db 31 c4
```

A.8.4. Расшифровка данных

Начало расшифровки данных AEAD-OCB с использованием сеансового ключа.

Данные HKDF

```
d2 02 07 02 06
```

Выход HKDF

```
45 12 f7 14 9d 86 33 41 52 7c 65 67 d5 bf fc 42
5f af 32 50 21 2f f9
```

Ключ сообщения

```
45 12 f7 14 9d 86 33 41 52 7c 65 67 d5 bf fc 42
```

Вектор инициализации

```
5f af 32 50 21 2f f9
```

Chunk #0

Nonce

```
5f af 32 50 21 2f f9 00 00 00 00 00 00 00 00
```

Дополнительные данные аутентификации

```
d2 02 07 02 06
```

Кусок (chunk) шифрованных данных

```
1b a9 eb d2 c0 ec 95 bf 56 9d 25 c9 99 ee 4a 3d
e1 70 58 f4 0d fa 8b 4c 68 2b e3 fb bb d7 b2 7e
b0 f5 9b b5 00 5f 80 c7 c6 f4 03 88 c3 0a d4 06
ab 05 13 dc d6
```

Расшифрованный chunk #0.

Пакет Literal Data со строкой Hello, world!

```
cb 13 62 00 00 00 00 00 48 65 6c 6c 6f 2c 20 77
6f 72 6c 64 21
```

Пакет дополнения

```
d5 0e c5 a2 93 07 29 91 62 81 47 d7 2c 8f 86 b7
```

Финальный тег аутентификации

Финальное значение nonce

```
5f af 32 50 21 2f f9 00 00 00 00 00 00 00 01
```

Финальные дополнительные данные аутентификации

```
d2 02 07 02 06 00 00 00 00 00 00 00 25
```

A.8.5. Полная последовательность шифрованного пакета X25519-AEAD-OCB

```
-----BEGIN PGP MESSAGE-----

wV0GIQYSyD8ecG9jCP4VGkF3Q6HwM3kOk+mXhIjR2zeNqZMIhRmHzxjV8bU/gXzO
WgBM85PMiVi93AZfJfhK9QmxfdNnZBjeo1VDeVZheQHgaVf7yopqR6W1FT6NOrfS
aQIHAgZhZBZTW+CwcW1g4FKlbExAf56zaw76/prQoN+bAzxpohup69LA7JW/Vp01
yZnuSj3hcFj0DfqLTGgr4/u717J+sPWbtQBfgMfG9AOIwwrUBqsFE9zW+f1zdlYo
bhF30A+IitsxxA==
-----END PGP MESSAGE-----
```

A.9. Пример шифрования и расшифровки AEAD-EAX

В этом примере шифруется текстовая строка «Hello, world!» с парольной фразой и AES-128 с шифрованием AEAD-EAX.

A.9.1. Пакет зашифрованного с симметричным ключом сеансового ключа v6

Ниже представлена последовательность октетов пакета.

```
0x0000  c3 40 06 1e 07 01 0b 03
0x0008  08 a5 ae 57 9d 1f c5 d8
0x0010  2b ff 69 22 4f 91 99 93
0x0018  b3 50 6f a3 b5 9a 6a 73
0x0020  cf f8 c5 ef c5 f4 1c 57
0x0028  fb 54 e1 c2 26 81 5d 78
0x0030  28 f5 f9 2c 45 4e b6 5e
0x0038  be 00 ab 59 86 c6 8e 6e
0x0040  7c 55
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  c3                      тип пакета SKESK
0x0001     40                   размер пакета
0x0002        06                v6 SKESK
0x0003           1e             размер до конца AEAD nonce
0x0004              07          шифр AES128
0x0005                 01       режим AEAD EAX
0x0006                    0b    размер S2K
0x0007                       03 тип S2K iterated+salted
0x0008  08                      хэш S2K SHA2-256
0x0009     a5 ae 57 9d 1f c5 d8 затравка S2K
0x0010  2b
0x0011     ff                   итерации S2K (65011712 октетов)
0x0012        69 22 4f 91 99 93 AEAD nonce
0x0018  b3 50 6f a3 b5 9a 6a 73
0x0020  cf f8
0x0022        c5 ef c5 f4 1c 57 зашифрованный сеансовый ключ
0x0028  fb 54 e1 c2 26 81 5d 78
0x0030  28 f5
0x0032        f9 2c 45 4e b6 5e тег AEAD
0x0038  be 00 ab 59 86 c6 8e 6e
0x0040  7c 55
```

A.9.2. Начало расшифровки AEAD-EAX для сеансового ключа

Выведенный ключ имеет вид

```
15 49 67 e5 90 aa 1f 92 3e 1c 0a c6 4c 88 f2 3d
```

данные HKDF

```
c3 06 07 01
```

Выход HKDF

```
2f ce 33 1f 39 dd 95 5c c4 1e 95 d8 70 c7 21 39
```

Аутентифицированные данные

```
c3 06 07 01
```

Nonce

```
69 22 4f 91 99 93 b3 50 6f a3 b5 9a 6a 73 cf f8
```

Расшифрованный сеансовый ключ

```
38 81 ba fe 98 54 12 45 9b 86 c3 6f 98 cb 9a 5e
```

A.9.3. Пример пакета SEIPD v2

Ниже представлена последовательность октетов пакета.

```
0x0000  d2 69 02 07 01 06 9f f9
0x0008  0e 3b 32 19 64 f3 a4 29
0x0010  13 c8 dc c6 61 93 25 01
0x0018  52 27 ef b7 ea ea a4 9f
0x0020  04 c2 e6 74 17 5d 4a 3d
0x0028  22 6e d6 af cb 9c a9 ac
0x0030  12 2c 14 70 e1 1c 63 d4
0x0038  c0 ab 24 1c 6a 93 8a d4
0x0040  8b f9 9a 5a 99 b9 0b ba
0x0048  83 25 de 61 04 75 40 25
0x0050  8a b7 95 9a 95 ad 05 1d
0x0058  da 96 eb 15 43 1d fe f5
0x0060  f5 e2 25 5c a7 82 61 54
0x0068  6e 33 9a
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  d2                      тип пакета SEIPD
0x0001     69                   размер пакета
0x0002        02                v2 SEIPD
0x0003           07             шифр AES128
0x0004              01          режим AEAD EAX
0x0005                 06       размер chunk (2^12 октетов)
0x0005                    9f f9 затравка (salt)
0x0008  0e 3b 32 19 64 f3 a4 29
0x0010  13 c8 dc c6 61 93 25 01
0x0018  52 27 ef b7 ea ea a4 9f
0x0020  04 c2 e6 74 17 5d
0x0026                    4a 3d chunk #0 зашифрованные данные
0x0028  22 6e d6 af cb 9c a9 ac
0x0030  12 2c 14 70 e1 1c 63 d4
0x0038  c0 ab 24 1c 6a 93 8a d4
0x0040  8b f9 9a 5a 99 b9 0b ba
0x0048  83 25 de
0x004b           61 04 75 40 25 chunk #0 тег AEAD
0x0050  8a b7 95 9a 95 ad 05 1d
0x0058  da 96 eb
0x005b           15 43 1d fe f5 финальный тег AEAD (#1)
0x0060  f5 e2 25 5c a7 82 61 54
0x0068  6e 33 9a
```

A.9.4. Расшифровка данных

Начало расшифровки данных AEAD-EAX с использованием сеансового ключа

Данные HKDF

```
d2 02 07 01 06
```

Выход HKDF

```
b5 04 22 ac 1c 26 be 9d dd 83 1d 5b bb 36 b6 4f
78 b8 33 f2 e9 4a 60 c0
```

Ключ сообщения

```
b5 04 22 ac 1c 26 be 9d dd 83 1d 5b bb 36 b6 4f
```

Вектор инициализации

```
78 b8 33 f2 e9 4a 60 c0
```

Chunk #0

Nonce

```
78 b8 33 f2 e9 4a 60 c0 00 00 00 00 00 00 00 00
```

Дополнительные данные аутентификации

```
d2 02 07 01 06
```

Расшифрованный chunk #0

Пакет Literal Data со строкой «Hello, world!»

```
cb 13 62 00 00 00 00 00 48 65 6c 6c 6f 2c 20 77
6f 72 6c 64 21
```

Пакет дополнения

```
d5 0e ae 5b f0 cd 67 05 50 03 55 81 6c b0 c8 ff
```

Финальный тег аутентификации

Финальное значение nonce

```
78 b8 33 f2 e9 4a 60 c0 00 00 00 00 00 00 00 01
```

Финальные дополнительные данные аутентификации

```
d2 02 07 01 06 00 00 00 00 00 00 00 25
```

A.9.5. Полная последовательность шифрованного пакета AEAD-EAX

```
-----BEGIN PGP MESSAGE-----

w0AGHgcBCwMIpa5XnR/F2Cv/aSJPkZmTs1Bvo7WaanPP+MxvxfQcV/tU4cImgV14
KPX5LEVOtl6+AKtZhsaObnxV0mkCBwEGn/kOOzIZZPOkKRPI3MZhkyUBUifvt+rq
pJ8EwuZ0F11KPSJu1q/LnKmsEiwUcOEcY9TAqyQcapOK1Iv5mlqZuQu6gyXeYQR1
QCWKt5Wala0FHdqW6xVDHf719eIlXKeCYVRuM5o=
-----END PGP MESSAGE-----
```

A.10. Пример шифрования и расшифровки AEAD-OCB

В это примере шифруется строка «Hello, world!» с парольной фразой и использованием AES-128 с шифром AEAD-OCB.

A.10.1. Пакет зашифрованного с симметричным ключом сеансового ключа v6

Ниже представлена последовательность октетов пакета.

```
0x0000  c3 3f 06 1d 07 02 0b 03
0x0008  08 56 a2 98 d2 f5 e3 64
0x0010  53 ff cf cc 5c 11 66 4e
0x0018  db 9d b4 25 90 d7 dc 46
0x0020  b0 72 41 b6 12 c3 81 2c
0x0028  ff fb ea 00 f2 34 7b 25
0x0030  64 11 23 f8 87 ae 60 d4
0x0038  fd 61 4e 08 37 d8 19 d3
0x0040  6c
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  c3                      тип пакета SKESK
0x0001     3f                   размер пакета
0x0002        06                v6 SKESK
0x0003           1d             размер до конца AEAD nonce
0x0004              07          шифр AES128
0x0005                 02       режим AEAD OCB
0x0006                    0b    размер S2K
0x0007                       03 тип S2K iterated+salted
0x0008  08                      хэш S2K SHA2-256
0x0009     56 a2 98 d2 f5 e3 64 затравка S2K
0x0010  53
0x0011    ff                    итерации S2K (65011712 октетов)
0x0012        cf cc 5c 11 66 4e AEAD nonce
0x0018  db 9d b4 25 90 d7 dc 46
0x0020  b0
0x0021     72 41 b6 12 c3 81 2c зашифрованный сеансовый ключ
0x0028  ff fb ea 00 f2 34 7b 25
0x0030  64
0x0031     11 23 f8 87 ae 60 d4 тег AEAD
0x0038  fd 61 4e 08 37 d8 19 d3
0x0040  6c
```

A.10.2. Начало расшифровки AEAD-OCB для сеансового ключа

Выведенный ключ имеет вид

```
e8 0d e2 43 a3 62 d9 3b 9d c6 07 ed e9 6a 73 56
```

Данные HKDF

```
c3 06 07 02
```

Выход HKDF

```
38 a9 b3 45 b5 68 0b b6 1b b6 5d 73 ee c7 ec d9
```

Аутентифицированные данные

```
c3 06 07 02
```

Nonce

```
cf cc 5c 11 66 4e db 9d b4 25 90 d7 dc 46 b0
```

Расшифрованный сеансовый ключ

```
28 e7 9a b8 23 97 d3 c6 3d e2 4a c2 17 d7 b7 91
```

A.10.3. Пример пакета SEIPD v2

Ниже представлена последовательность октетов пакета.

```
0x0000  d2 69 02 07 02 06 20 a6
0x0008  61 f7 31 fc 9a 30 32 b5
0x0010  62 33 26 02 7e 3a 5d 8d
0x0018  b5 74 8e be ff 0b 0c 59
0x0020  10 d0 9e cd d6 41 ff 9f
0x0028  d3 85 62 75 80 35 bc 49
0x0030  75 4c e1 bf 3f ff a7 da
0x0038  d0 a3 b8 10 4f 51 33 cf
0x0040  42 a4 10 0a 83 ee f4 ca
```

```
0x0048  1b 48 01 a8 84 6b f4 2b
0x0050  cd a7 c8 ce 9d 65 e2 12
0x0058  f3 01 cb cd 98 fd ca de
0x0060  69 4a 87 7a d4 24 73 23
0x0068  f6 e8 57
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  d2                      тип пакета SEIPD
0x0001     69                   размер пакета
0x0002        02                v2 SEIPD
0x0003           07             шифр AES128
0x0004              02          режим AEAD OCB
0x0005                 06       размер chunk (2^12 октетов)
0x0006                    20 a6 затравка (salt)
0x0008  61 f7 31 fc 9a 30 32 b5
0x0010  62 33 26 02 7e 3a 5d 8d
0x0018  b5 74 8e be ff 0b 0c 59
0x0020  10 d0 9e cd d6 41
0x0026                    ff 9f  chunk #0 шифрованные данные
0x0028  d3 85 62 75 80 35 bc 49
0x0030  75 4c e1 bf 3f ff a7 da
0x0038  d0 a3 b8 10 4f 51 33 cf
0x0040  42 a4 10 0a 83 ee f4 ca
0x0048  1b 48 01
0x004b           a8 84 6b f4 2b  chunk #0 тег аутентификации
0x0050  cd a7 c8 ce 9d 65 e2 12
0x0058  f3 01 cb
0x005b           cd 98 fd ca de  финальный тег AEAD (#1)
0x0060  69 4a 87 7a d4 24 73 23
0x0068  f6 e8 57
```

A.10.4. Расшифровка данных

Начало расшифровки данных AEAD-OCB с использованием сеансового ключа

Данные HKDF

```
d2 02 07 02 06
```

Выход HKDF

```
71 66 2a 11 ee 5b 4e 08 14 4e 6d e8 83 a0 09 99
eb de 12 bb 57 0d cf
```

Ключ сообщения

```
71 66 2a 11 ee 5b 4e 08 14 4e 6d e8 83 a0 09 99
```

Вектор инициализации

```
eb de 12 bb 57 0d cf
```

Chunk #0

Nonce

```
eb de 12 bb 57 0d cf 00 00 00 00 00 00 00 00
```

Дополнительные данные аутентификации

```
d2 02 07 02 06
```

Расшифрованый chunk #0

Пакет Literal Data со строкой содержимого «Hello, world!»

```
cb 13 62 00 00 00 00 00 48 65 6c 6c 6f 2c 20 77
6f 72 6c 64 21
```

Пакет дополнения

```
d5 0e ae 6a a1 64 9b 56 aa 83 5b 26 13 90 2b d2
```

Финальный тег аутентификации

Финальное значение nonce

```
eb de 12 bb 57 0d cf 00 00 00 00 00 00 00 01
```

Финальные дополнительные данные аутентификации

```
d2 02 07 02 06 00 00 00 00 00 00 00 25
```

A.10.5. Полная последовательность шифрованного пакета AEAD-OCB

```
-----BEGIN PGP MESSAGE-----

wz8GHQcCCwMIVqKY0vXjZFP/z8xcEWZO2520JZDX3EawckG2EsOBLP/76gDyNHsl
ZBEj+IeuYNT9YU4IN9gZ02zSaQIHAgYgpmH3MfyaMDK1YjMmAn46XY21dI6+/wsM
WRDQns3WQf+f04VidYA1vEl1TOG/P/+n2tCjuBBPUTPPQqQQCoPu9MobSAGohGv0
K82nyM6dZeIS8wHLzZj9yt5pSod61CRzI/boVw==
-----END PGP MESSAGE-----
```

A.11. Пример шифрования и расшифровки AEAD-GCM

В это примере шифруется строка «Hello, world!» с парольной фразой и использованием AES-128 с шифром AEAD-GCM.

A.11.1. Пакет зашифрованного с симметричным ключом сеансового ключа v6

Ниже представлена последовательность октетов пакета.

```
0x0000  c3 3c 06 1a 07 03 0b 03
0x0008  08 e9 d3 97 85 b2 07 00
0x0010  08 ff b4 2e 7c 48 3e f4
0x0018  88 44 57 cb 37 26 b9 b3
0x0020  db 9f f7 76 e5 f4 d9 a4
0x0028  09 52 e2 44 72 98 85 1a
0x0030  bf ff 75 26 df 2d d5 54
0x0038  41 75 79 a7 79 9f
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  c3                      тип пакета SKESK
0x0001     3c                   размер пакета
0x0002        06                v6 SKESK
0x0003           1a             размер до конца AEAD nonce
0x0004              07          шифр AES128
0x0005                 03       режим AEAD GCM
0x0006                    0b    length of S2K
0x0007                       03 тип S2K iterated+salted
0x0008  08                      хэш S2K SHA2-256
0x0009     e9 d3 97 85 b2 07 00 затравка S2K
0x0010  08
0x0011     ff                   итерации S2K (65011712 октетов)
0x0012        b4 2e 7c 48 3e f4 AEAD nonce
0x0018  88 44 57 cb 37 26
0x001e                    b9 b3 зашифрованный сеансовый ключ
0x0020  db 9f f7 76 e5 f4 d9 a4
0x0028  09 52 e2 44 72 98
0x002e                     85 1a  тег AEAD
0x0030  bf ff 75 26 df 2d d5 54
0x0038  41 75 79 a7 79 9f
```

A.11.2. Начало расшифровки AEAD-GCM для сеансового ключа

Выведенный ключ имеет вид

```
25 02 81 71 5b ba 78 28 ef 71 ef 64 c4 78 47 53
```

Данные HKDF

```
c3 06 07 03
```

Выход HKDF

```
7a 6f 9a b7 f9 9f 7e f8 db ef 84 1c 65 08 00 f5
```

Аутентифицированные данные

```
c3 06 07 03
```

Nonce

```
b4 2e 7c 48 3e f4 88 44 57 cb 37 26
```

Расшифрованный сеансовый ключ

```
19 36 fc 85 68 98 02 74 bb 90 0d 83 19 36 0c 77
```

A.11.3. Пример пакета SEIPD v2

Ниже представлена последовательность октетов пакета.

```
0x0000  d2 69 02 07 03 06 fc b9
0x0008  44 90 bc b9 8b bd c9 d1
0x0010  06 c6 09 02 66 94 0f 72
0x0018  e8 9e dc 21 b5 59 6b 15
0x0020  76 b1 01 ed 0f 9f fc 6f
0x0028  c6 d6 5b bf d2 4d cd 07
0x0030  90 96 6e 6d 1e 85 a3 00
0x0038  53 78 4c b1 d8 b6 a0 69
0x0040  9e f1 21 55 a7 b2 ad 62
0x0048  58 53 1b 57 65 1f d7 77
0x0050  79 12 fa 95 e3 5d 9b 40
0x0058  21 6f 69 a4 c2 48 db 28
0x0060  ff 43 31 f1 63 29 07 39
0x0068  9e 6f f9
```

Те же данные с разбивкой по октетам и семантикой показаны ниже.

```
0x0000  d2                      тип пакета SEIPD
0x0001     69                   размер пакета
0x0002        02                v2 SEIPD
0x0003           07             шифр AES128
0x0004              03          режим AEAD GCM
0x0005                 06       размер квска (2^12 октетов)
0x0006                    fc b9 затравка (salt)
0x0008  44 90 bc b9 8b bd c9 d1
0x0010  06 c6 09 02 66 94 0f 72
0x0018  e8 9e dc 21 b5 59 6b 15
0x0020  76 b1 01 ed 0f 9f
0x0026                    fc 6f  chunk #0 зашифрованные данные
0x0028  c6 d6 5b bf d2 4d cd 07
0x0030  90 96 6e 6d 1e 85 a3 00
0x0038  53 78 4c b1 d8 b6 a0 69
0x0040  9e f1 21 55 a7 b2 ad 62
0x0048  58 53 1b
0x004b           57 65 1f d7 77  chunk #0 тег аутентификации
```

```
0x0050  79 12 fa 95 e3 5d 9b 40
0x0058  21 6f 69
0x005b           a4 c2 48 db 28  финальный тег AEAD (#1)
0x0060  ff 43 31 f1 63 29 07 39
0x0068  9e 6f f9
```

A.11.4. Расшифровка данных

Начало расшифровки AEAD-GCM с использованием сеансового ключа

Данные HKDF

```
d2 02 07 03 06
```

Выход HKDF

```
ea 14 38 80 3c b8 a4 77 40 ce 9b 54 c3 38 77 8d
4d 2b dc 2b
```

Ключ сообщения

```
ea 14 38 80 3c b8 a4 77 40 ce 9b 54 c3 38 77 8d
```

Вектор инициализации

```
4d 2b dc 2b
```

Chunk #0

Nonce

```
4d 2b dc 2b 00 00 00 00 00 00 00 00
```

Дополнительные данные аутентификации

```
d2 02 07 03 06
```

Расшифрованый chunk #0

Пакет Literal Data со строкой содержимого «Hello, world!»

```
cb 13 62 00 00 00 00 00 48 65 6c 6c 6f 2c 20 77
6f 72 6c 64 21
```

Пакет дополнения

```
d5 0e 1c e2 26 9a 9e dd ef 81 03 21 72 b7 ed 7c
```

Финальный тег аутентификации

Финальное значение nonce

```
4d 2b dc 2b 00 00 00 00 00 00 00 01
```

Финальные дополнительные данные аутентификации

```
d2 02 07 03 06 00 00 00 00 00 00 00 25
```

A.11.5. Полная последовательность шифрованного пакета AEAD-GCM

```
-----BEGIN PGP MESSAGE-----

wzwGGgcDCwMI6dOXhbIHAAj/tC58SD70iERXyzcmubPbn/d25fTZpAlS4kRymIUa
v/91Jt8t1VRBdXmneZ/SaQIHAwb8uUSQvLmLvcnRBsYJAmaUD3LontwhtVlrFXax
Ae0Pn/xvxtZbv9JNzQeQlm5tHoWjAFN4TLHYtqBpnvEhVaeyrWJYUxtXZR/Xd3kS
+pXjXZtAIW9ppMJI2yj/QzHxYykHOZ5v+Q==
-----END PGP MESSAGE-----
```

A.12. Пример сообщений, зашифрованных с помощью Argon2

Эти сообщения содержат литеральные данные «Hello, world!», зашифрованные с использованием v1 SEIPD с Argon2 и паролем «password» при разных размерах сеансового ключа. В каждом примере выбирается один симметричный шифр для пакета v4 SKESK и v1 SEIPD. Параметры Argon2 имеют значения t = 1, p = 4, m = 21.

A.12.1. V4 SKESK, использующий Argon2 с AES-128

```
-----BEGIN PGP MESSAGE-----
Comment: Encrypted using AES with 128-bit key
Comment: Session key: 01FE16BBACFD1E7B78EF3B865187374F

wycEBwScUvg8J/leUNU1RA7N/zE2AQQVnlL8rSLPP5VlQsunlO+EcxHSPgGYGKY+
YJz4u6F+DDlDBOr5NRQXt/KJIf4m4mOlKyC/uqLbpnLJZMnTq3o79GxBTdIdOzhH
XfA3pqV4mTzF
-----END PGP MESSAGE-----
```

A.12.2. V4 SKESK, использующий Argon2 с AES-192

```
-----BEGIN PGP MESSAGE-----
Comment: Encrypted using AES with 192-bit key
Comment: Session key: 27006DAE68E509022CE45A14E569E91001C2955…
Comment: Session key: ...AF8DFE194

wy8ECAThTKxHFTRZGKli3KNH4UP4AQQVhzLJ2va3FG8/pmpIPd/H/mdoVS5VBLLw
F9I+AdJ1Sw56PRYiKZjCvHg+2bnq02s33AJJoyBexBI4QKATFRkyez2gldJldRys
LVg77Mwwfgl2n/d572WciAM=
-----END PGP MESSAGE-----
```

A.12.3. V4 SKESK, использующий Argon2 с AES-256

```
-----BEGIN PGP MESSAGE-----
Comment: Encrypted using AES with 256-bit key
Comment: Session key: BBEDA55B9AAE63DAC45D4F49D89DACF4AF37FEF…
```

```
Comment: Session key: ...C13BAB2F1F8E18FB74580D8B0

wzcECQS4eJUgIG/3mcaILEJFpmJ8AQQVnZ9l7KtagdClm9UaQ/Z6M/5roklSGpGu
623YmaXezGj80j4B+Ku1sgTdJo87X1Wrup7l0wJypZls21Uwd67m9koF60eefH/K
95D1usliXOEm8ayQJQmZrjf6K6v9PWwqMQ==
-----END PGP MESSAGE-----
```

Приложение B. Обновления (реализации RFC 4880 и 6637)

В этом приложении представлена краткая ненормативная сводка существенных добавлений и исключений для [RFC4880] и [RFC6637]. Это предназначено для оказания помощи разработчикам, расширяющим реализации этих спецификаций в соответствии с данным документом. Криптоалгоритмы с пометкой MTI обязательны для реализации.

- Алгоритмы подписи с открытым ключом:
  - Ed25519 (параграфы 5.5.5.9 and 5.2.3.4) - MTI
  - Ed448 (параграфы 5.5.5.10 and 5.2.3.5)
  - EdDSALegacy с Ed25519Legacy (параграфы 5.5.5.5 and 5.2.3.3)
  - ECDSA с кривыми Brainpool (параграф 9.2)
- Алгоритмы шифрования с открытым ключом:
  - X25519 (параграфы 5.5.5.7 and 5.1.6) - MTI
  - X448 (параграфы 5.5.5.8 and 5.1.7)
  - ECDH с Curve25519Legacy (параграф 9.2)
  - ECDH с кривыми Brainpool (параграф 9.2)
- Шифрования AEAD:
  - V2 SEIPD (параграф 5.13.2)
  - режимы AEAD:
    - OCB (параграф 5.13.4) - MTI
    - EAX (параграф 5.13.3)
    - GCM (параграф 5.13.5)
  - V6 PKESK (параграф 5.1.2)
  - V6 SKESK (параграф 5.3.2)
  - Субпакет подписи Features: добавлен флаг для v2 SEIPD (параграф 5.2.3.32)
  - Субпакет Signature: предпочтительные шифры AEAD (параграф 5.2.3.15)
  - Шифрование секретного ключа: AEAD «октет использования S2K» (параграфы 3.7.2 and 5.5.3)
- Ключи и подписи версии 6:
  - Открытые ключи версии 6 (параграф 5.5.2.3)
  - 6 Fingerprint и Key ID версии 6 (параграф 5.5.4.3)
  - Секретные ключи версии 6 (параграф 5.5.3)
  - Подписи версии 6 (параграф 5.2.3)
  - Одноразовые подписи версии 6 (параграф 5.4)
- Структура сертификата (Transferable Public Key):
  - Предпочтения субпакетов в подписях Direct Key (параграф 5.2.3.10)
  - Самопроверяющий сертификат отзыва (параграф 10.1.2)
  - User ID является необязательным (параграф 10.1.1)
- S2K: Argon2 (параграф 3.7.1.4)
- Субпакет: Оттиск предусмотренного получателя (параграф 5.2.3.36)
- Алгоритмы подписи: SHA3-256 и SHA3-512 (параграф 9.5)
- Пакет Padding (параграф 5.14)
- структура сообщения: критичность пакетов (параграф 4.3)
- Отмены:
  - Алгоритмы с открытым ключом:
    - Избегать слабых ключей RSA (параграф 12.4)
    - Избегать DSA (параграф 12.5)
    - Избегать ElGamal (параграфы 12.6 and 5.1.4)
    - Для ключей версии 6 избегать EdDSA25519Legacy и Curve25519Legacy (параграф 9.2)
  - Алгоритмы подписи:
    - Избегать MD5, SHA1, RIPEMD160 (параграф 9.5)
  - Алгоритмы подписи с симметричным ключом:
    - Избегать IDEA, TripleDES, CAST5 (параграф 9.3)
  - Спецификатор S2K:
    - Избегать Simple S2K (параграф 3.7.1.1)
  - Защита секретных ключей (a.k.a. S2K Usage):
    - Избегать MalleableCFB (параграф 3.7.2.1)
  - Типы пакетов:
    - Избегать Symmetrically Encrypted Data (параграфы 5.7 and 13.7)
  - Метаданные пакета Literal Data:
    - Избегать полей Filename и Date (параграф 5.9)
    - Избегать специального имени файла _CONSOLE (параграф 5.9.1)
  - Версии пакетов:
    - Избегать открытых ключей версии 3 (параграф 5.5.2.1)
    - Избегать подписей версии 3 (параграф 5.2)
  - Типы подписей:
    - Избегать резервного идентификатора типа подписи 0xFF (параграфы 5.2.1.16 и 5.2.4.1)
  - Субпакеты Signature:
    - Для подписей версии 6 избегать Issuer Key ID (параграф 5.2.3.12)
    - Избегать Revocation Key (параграф 5.2.3.23)
  - ASCII Armor:
    - Игнорировать отсутствие CRC (параграф 6.1)
    - Не выдавать заголовок Armor Version (параграф 6.2.2.1)

- Схема с открытой подписью:
  - Игнорировать; избегать выдачи нетребуемого Hash: заголовки (параграф 6.2.2.3)
  - Отвергать подписи в открытом виде с недействительным Hash: заголовки (параграф 6.2.2.3) или иные Armor Header (параграф 7.1)

B.1. Изменения в терминологии

Некоторые термины, применявшиеся в прежних версиях спецификации OpenPGP, в этом документе заменены более подходящими. Ниже перечислены термины, использовавшиеся в прежних версиях.

- Radix-64 для обозначения кодирования OpenPGP ASCII Armor base64 (раздел 6).
- Old packet format (старый формат пакета) для формата пакетов Legacy [RFC2440] (параграф 4.2.2) .
- New packet format (новый формат пакета) для формата пакетов OpenPGP (параграф 4.2.1), введённого в [RFC2440].
- Термин сертификат мог относиться к разным элементам, но в данном документе он означает лишь Transferable Public Key (переносимый открытый ключ).
- Preferred Symmetric Algorithms (предпочтительные симметричные алгоритмы) для субпакета Preferred Symmetric Ciphers for v1 SEIPD (параграф 5.2.3.14).
- Modification Detection Code или MDC изначально описан как отдельный пакет (Packet Type ID 19), а соответствующий ему флаг в субпакет подписи Features (параграф 5.2.3.32) назывался Modification Detection (обнаружение изменения). Сейчас это внутренняя часть v1 SEIPD (параграф 5.13.1), а соответствующий флаг называется Version 1 Symmetrically Encrypted and Integrity Protected Data packet (пакет данных с симметричным шифрованием и защитой целостности версии 1).
- Packet Tag для Packet Type ID (идентификатор типа пакета, раздел 5), а иногда для кодированного Packet Type ID (параграф 4.2).

Приложение C. Ошибки, устранённые этим документом

Ниже перечислены подтверждённые ошибки, исправленные этим документом.

- [Errata-2199] - корректировка октета хэш-шифр S2K
- [Errata-2200] - отсутствие неявного использования коррекции IDEA
- [Errata-2206] - аббревиатура PKESK
- [Errata-2208] - уточнение по части владельца ключа
- [Errata-2214] - уточнение хэширования подписи
- [Errata-2216] - применение самоподписи для коррекции идентификатора пользователя
- [Errata-2219] - уточнение по части хранилища шифрованных сеансовых ключей
- [Errata-2222] - уточнение по части простого хэша (можно-должно)
- [Errata-2226] - уточнение по части естественного завершения строк (следует)
- [Errata-2234] - уточнение Radix-64/base64
- [Errata-2235] - уточнение по части порядка сортировки ASCII/UTF-8
- [Errata-2236] - уточнение по части составления пакета
- [Errata-2238] - уточнение по части пакетов субключей после всех пакетов User ID
- [Errata-2240] - уточнение по части удаления субключа
- [Errata-2242] - уточнение для переменной mL/emLen
- [Errata-2243] - уточнение для вектора инициализации (IV) в режиме CFB
- [Errata-2270] - корректировка последовательности октетов SHA-224
- [Errata-2271] - корректировка Radix-64
- [Errata-3298] - корректировка подписей отзыва ключа
- [Errata-5491] - исправление кода C для определения CRC24_POLY
- [Errata-7545] - исправление шестнадцатеричных значений столбца Armor Header
- [Errata-7889] - корректировка подписи/сертификации

Благодарности

Спасибо команде OpenPGP Design за работу над этим документом и его подготовку для рабочей группы. Команда включала Stephen Farrell, Daniel Kahn Gillmor, Daniel Huigens, Jeffrey Lau, Yutaka Niibe, Justus Winter, Paul Wouters.

Спасибо Werner Koch за работу над rfc4880bis и Andrey Jivsov за работу над [RFC6637].

Этот документ опирается на прежние работы ряда других авторов, включая Derek Atkins, Charles Breed, Dave Del Torto, Marc Dyksterhouse, Gail Haspert, Gene Hoffman, Paul Hoffman, Ben Laurie, Raph Levien, Colin Plumb, Will Price, Daphne Shaw, William Stallings, Mark Weaver, Philip R. Zimmermann.

Адреса авторов

Paul Wouters (editor)
Aiven
Email: paul.wouters@aiven.io

Daniel Huigens
Proton AG
Email: d.huigens@protonmail.com

Justus Winter
Sequoia PGP
Email: justus@sequoia-pgp.org

Yutaka Niibe
FSIJ
Email: gniibe@fsij.org

Перевод на русский язык

Николай Малых
nmalykh@protokols.ru