ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Р 1323565.1.020—2018

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.2)

Издание официальное

Москва
Стандартинформ
2018

gost.e-ecolog.ru

Предисловие

1 РАЗРАБОТАНЫ Обществом с ограниченной ответственностью «КРИПТО-ПРО» (ООО «КРИПТО-ПРО»)

2 ВНЕСЕНЫ Техническим комитетом по стандартизации ТК 26 «Криптографическая защита информации»

3 УТВЕРЖДЕНЫ И ВВЕДЕНЫ В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 22 августа 2018 г. № 511-ст

4 ВВЕДЕНЫ ВПЕРВЫЕ

*Правила применения настоящих рекомендаций установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящим рекомендациям публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящих рекомендаций соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© Стандартинформ, оформление, 2018

Настоящие рекомендации не могут быть полностью или частично воспроизведены, тиражированы и распространены в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения . . . 1
2 Нормативные ссылки . . . 1
3 Термины, определения и обозначения . . . 2
3.1 Термины и определения . . . 2
3.2 Обозначения . . . 2
4 Иерархия информационного обмена в протоколе TLS . . . 3
5 Протокол Record . . . 5
5.1 Состояние соединения . . . 5
5.2 Формирование записи . . . 6
6 Протокол Handshake . . . 9
6.1 Схема взаимодействия сторон . . . 10
6.2 Формат сообщений протокола Handshake . . . 12
6.3 Сообщения приветствия . . . 12
6.4 Выработка и подтверждение ключа . . . 16
7 Протокол Change Cipher Spec . . . 20
8 Протокол Alert . . . 20
8.1 Оповещения закрытия соединения . . . 21
8.2 Оповещения об ошибках . . . 22
9 Протокол Application Data . . . 23
10 Криптонаборы . . . 23
10.1 Сертификаты сторон . . . 24
10.2 Блочный шифр . . . 24
10.3 Алгоритм выработки кода аутентификации сообщения . . . 24
10.4 Алгоритм шифрования сообщений . . . 24
10.5 Максимальное количество записей в соединении . . . 24
10.6 Параметры ключевого дерева . . . 24
11 Вопросы безопасности . . . 25
Приложение А (справочное) Контрольные примеры работы алгоритма TLSTREE . . . 26
А.1 TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC . . . 26
А.2 TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC . . . 27
Приложение Б (справочное) Контрольные примеры для работы протокола Record . . . 30
Б.1 TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC . . . 30
Б.2 TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC . . . 32
Приложение В (справочное) Контрольные примеры для работы протокола TLS . . . 36
В.1 TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC . . . 36
В.2 TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC . . . 47
Библиография . . . 62

Введение

Настоящие рекомендации содержат описание протокола безопасности транспортного уровня версии 1.2 (TLS 1.2), с криптонаборами на основе алгоритмов блочного шифрования «Магма» и «Кузнечик», описанных в ГОСТ Р 34.12—2015.

Необходимость разработки настоящего документа вызвана потребностью в обеспечении совместимости реализаций протокола TLS с использованием российских государственных криптографических стандартов.

Примечание — Основная часть настоящих рекомендаций дополнена приложениями А—В.

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.2)

Information technology. Cryptographic data security.
The use of the russian cryptographic algorithms in the transport layer security protocol (TLS 1.2)

Дата введения — 2019—02—01

1 Область применения

Описанный в данных рекомендациях протокол предназначен для установления защищенного соединения между клиент-серверными приложениями в сети Интернет с использованием алгоритмов, определяемых российскими государственными криптографическими стандартами, для обеспечения аутентификации сторон, конфиденциальности и целостности информации, передаваемой по каналу связи, в котором допускается присутствие активного противника. Протокол может применяться для обеспечения защиты каналов связи при обработке информации, не содержащей сведений, составляющих государственную тайну.

2 Нормативные ссылки

В настоящих рекомендациях использованы нормативные ссылки на следующие стандарты:

ГОСТ Р 34.10—2012 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи

ГОСТ Р 34.11—2012 Информационная технология. Криптографическая защита информации. Функция хэширования

ГОСТ Р 34.12—2015 Информационная технология. Криптографическая защита информации. Блочные шифры

ГОСТ Р 34.13—2015 Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров

ГОСТ Р ИСО/МЭК 9594-8 Информационная технология. Взаимосвязь открытых систем. Справочник. Часть 8. Основы аутентификации

Р 50.1.113—2016 Информационная технология. Криптографическая защита информации. Криптографические алгоритмы, сопутствующие применению алгоритмов электронной цифровой подписи и функции хэширования

Р 1323565.1.017—2018 Информационная технология. Криптографическая защита информации. Криптографические алгоритмы, сопутствующие применению алгоритмов блочного шифрования

Примечание — При пользовании настоящими рекомендациями целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год. Если заменен

Издание официальное

ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящих рекомендаций в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, применяется в части, не затрагивающей эту ссылку.

3 Термины, определения и обозначения

3.1 Термины и определения

В настоящих рекомендациях применены следующие термины с соответствующими определениями:

3.1.1 **криптонабор** (cipher suite): Набор криптографических алгоритмов и их параметров, определяющий работу протокола TLS в рамках соответствующей данному криптонабору сессии.

3.1.2 **согласованный криптонабор:** Криптонабор, соответствующий идентификатору криптонабора, согласованному сторонами взаимодействия в процессе выполнения протокола Handshake, описанного в разделе 6.

3.1.3 **открытый ключ сервера:** Ключ, равный ключу проверки подписи, хранящемуся в сертификате сервера.

3.1.4 **закрытый ключ сервера:** Ключ, равный ключу подписи сервера, соответствующему ключу проверки подписи, хранящемуся в сертификате сервера.

Примечание — В настоящих рекомендациях термины «открытый ключ сервера», «закрытый ключ сервера» используются вместо терминов «ключ проверки подписи сервера», «ключ подписи сервера» в целях конкретизации способа их использования и указывают на то, что данные ключи не используются в протоколе TLS 1.2 для формирования подписи, а участвуют в выработке общего ключа при работе протокола Handshake (см. раздел 6).

3.2 Обозначения

В настоящих рекомендациях использованы следующие обозначения:

$x \bmod y$ — остаток от деления целого числа x на целое число y;

B_s — множество байтовых строк длины s, $s \geq 0$. Строка $b = (b_1,\ldots,b_s)$ принадлежит множеству B_s, если $b_1,\ldots,b_s \in \{0,\ldots,255\}$. При $s = 0$ множество B_s состоит из единственной пустой строки длины 0;

$|$ — конкатенация двух байтовых строк; для двух строк $a = (a_1,\ldots,a_{s1}) \in B_{s1}$, $b = (b_1,\ldots,b_{s2}) \in B_{s2}$ их конкатенацией $a \mid b$ называется строка $c = (a_1,\ldots,a_{s1},b_1,\ldots,b_{s2}) \in B_{s1+s2}$;

$b[i..j]$ — строка $b[i..j] = (b_i,b_{i+1},\ldots,b_j) \in B_{j-i+1}$, где $1 \leq i \leq j \leq s$ и $b = (b_1,\ldots,b_s) \in B_s$;

$INT(b)$ — **число** $INT(b) = 256^{s-1} \cdot b_1 + \cdots + 256 \cdot b_{s-1} + b_s$, где $b = (b_1,\ldots,b_s) \in B_s$;

$STR_s(r)$ — **строка** $STR_s(r) = (b_1,\ldots,b_s) \in B_s$, соответствующая числу $r = 256^{s-1} \cdot b_1 + \cdots + 256 \cdot b_{s-1} + b_s \leq 256^s - 1$;

$str_s(r)$ — строка $str_s(r) = (b_s,\ldots,b_1) \in B_s$, соответствующая числу $r = 256^{s-1} \cdot b_1 + \cdots + 256 \cdot b_{s-1} + b_s \leq 256^s - 1$;

$LMB_t(b)$ — строка $LMB_t(b) = (b_1,\ldots,b_t) \in B_t$, соответствующая строке $b = (b_1,\ldots,b_s) \in B_s$, $1 \leq t \leq s$.

n — параметр алгоритма блочного шифрования, называемый длиной блока; в рамках данного документа измеряется в байтах;

k — параметр алгоритма блочного шифрования, называемый длиной ключа; в рамках данного документа измеряется в байтах;

$ENC(K,IV,M)$ — алгоритм шифрования сообщения M на ключе K с помощью вектора инициализации IV, который задается выбранным криптонабором;

$MAC(K,M)$ — алгоритм вычисления кода аутентификации сообщения M на ключе K, который задается выбранным криптонабором;

q — порядок подгруппы группы точек эллиптической кривой, соответствующей открытому ключу сервера;

P — точка эллиптической кривой порядка q, соответствующей открытому ключу сервера;

O — нулевая точка эллиптической кривой;

Q_C — ключ проверки подписи, хранящийся в сертификате клиента;

k_C — ключ подписи клиента, соответствующий ключу Q_C;

Q_S — открытый ключ сервера;

k_S — закрытый ключ сервера;

r_C — байтовая строка со случайными данными, соответствующая полю ClientHello.random, описанному в 6.3.2;

r_S — байтовая строка со случайными данными, соответствующая полю ServerHello.random, описанному в 6.3.3;

$SIGN_K$ — алгоритм подписи на ключе K, описанный в 6.4.6;

HASH — хэш-функция ГОСТ Р 34.11—2012 с длиной выхода 32 байта (256 бит);

KEG — алгоритм генерации ключей экспорта, описанный в 6.4.5.1;

PRF — псевдослучайная функция PRF_TLS_GOSTR3411_2012_256, описанная в Р 50.1.113—2016;

*KExp*15 — алгоритм экспорта ключей, описанный в Р 1323565.1.017—2018, использующий блочных шифр, задающийся выбранным криптонабором;

*KImp*15 — алгоритм импорта ключей, описанный в Р 1323565.1.017—2018, использующий блочных шифр, задающийся выбранным криптонабором;

& — операция побитовой конъюнкции (побитовое «И»);

HM — переменная, хранящая конкатенацию всех пересланных и полученных сообщений протокола Handshake, начиная с сообщения приветствия ClientHello и заканчивая сообщением (но не включая его), при формировании которого эта переменная используется; при этом сообщение запроса приветствия HelloRequest, сообщения протокола Change Cipher Spec и сообщения протокола Alert не включаются в переменную *HM*.

Примечание 1 — Для описания протокола в настоящих рекомендациях используется общепринятый язык представления (далее — язык представления TLS), описанный в [1].

Примечание 2 — В настоящих рекомендациях все строковые константы приводятся в кавычках и представляются в кодировке ASCII.

Примечание 3 — В настоящих рекомендациях в целях сохранения терминологической преемственности с действующими отечественными нормативными документами и опубликованными научно-техническими изданиями установлено, что термины «хэш-функция», «криптографическая хэш-функция», «функция хэширования» и «криптографическая функция хэширования» являются синонимами.

Примечание 4 — В настоящих рекомендациях в целях сохранения терминологической преемственности с действующими отечественными нормативными документами и опубликованными научно-техническими изданиями установлено, что термины «электронная подпись», «цифровая подпись», «электронная цифровая подпись» и «подпись» являются синонимами.

Примечание 5 — В настоящих рекомендациях используются обозначения параметров эллиптических кривых в соответствии с ГОСТ Р 34.10—2012 (раздел 5).

4 Иерархия информационного обмена в протоколе TLS

Протокол TLS состоит из двух уровней: нижнего и верхнего. На нижнем уровне находится протокол Record, работающий поверх некоторого транспортного протокола (например, TCP) с гарантированной доставкой пакетов данных, который обеспечивает доставку сообщений с сохранением их очередности, отсутствием потерь и дублирований. Поверх протокола Record, в свою очередь, работают следующие протоколы: протокол Handshake, протокол Alert, протокол Change Cipher Spec и протокол Application Data.

Схема обмена данными в протоколе TLS изображена на рисунке 1.

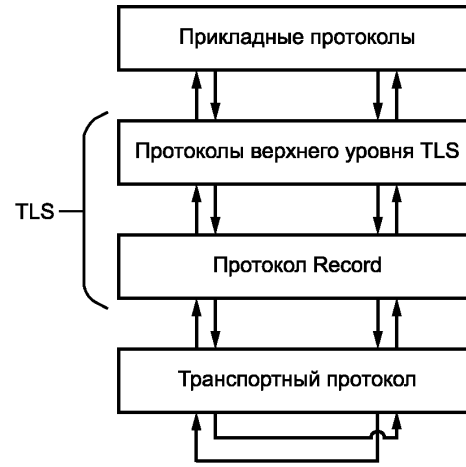


Рисунок 1 — Схема обмена данными в протоколе TLS

Основное назначение протокола TLS — создание защищенного канала связи между двумя взаимодействующими сторонами, то есть обеспечение следующих свойств:

- аутентификация сторон: криптонаборы, описанные в настоящих рекомендациях, предоставляют возможность односторонней или двусторонней аутентификации сторон. Аутентификация клиента осуществляется за счет проверки подписи клиента; аутентификация сервера осуществляется за счет подтверждения обладания общим секретом с помощью закрытого ключа, соответствующего сертификату сервера;
- конфиденциальность: обеспечивается за счет шифрования передаваемой информации;
- целостность: обеспечивается за счет использования кодов аутентификации передаваемых сообщений.

Иерархия информационного обмена протокола TLS включает в себя сессии, соединения, поток сообщений различных типов, который разбивается на записи. В одной сессии может быть реализовано несколько соединений, произвольно разнесенных по времени.

Каждая сессия характеризуется следующими параметрами безопасности, которые согласовываются в ходе работы протокола Handshake (см. раздел 6) и остаются неизменными для каждого соединения в рамках данной сессии:

- идентификатор сессии;
- сертификат сервера;
- сертификат клиента (в случае двусторонней аутентификации);
- криптонабор;
- общее сессионное секретное значение *MS* (формируется клиентом и сервером в ходе работы протокола Handshake, соответствующего полной схеме обмена сообщениями).

Каждое соединение характеризуется следующими параметрами безопасности, которые согласовываются в ходе протокола Handshake (см. раздел 6):

- строка со случайными байтами r_C, задаваемая клиентом;
- строка со случайными байтами r_S, задаваемая сервером;
- ключ клиента для вычисления кода аутентификации сообщения $K_{MAC,C}^{write}$;
- ключ клиента для проверки кода аутентификации сообщения $K_{MAC,C}^{read}$;
- ключ сервера для вычисления кода аутентификации сообщения $K_{MAC,S}^{write}$;
- ключ сервера для проверки кода аутентификации сообщения $K_{MAC,S}^{read}$;
- ключ клиента для зашифрования данных $K_{ENC,C}^{write}$;
- ключ клиента для расшифрования данных $K_{ENC,C}^{read}$;
- ключ сервера для зашифрования данных $K_{ENC,S}^{write}$;
- ключ сервера для расшифрования данных $K_{ENC,S}^{read}$;

- вектор инициализации клиента для зашифрования данных IV_C^{write};
- вектор инициализации клиента для расшифрования данных IV_C^{read};
- вектор инициализации сервера для зашифрования данных IV_S^{write};
- вектор инициализации сервера для расшифрования данных IV_S^{read}.

Примечание 1 — При первичном открытии сессии во время выполнения протокола Handshake вырабатываются как параметры сессии, так и параметры соединения.

Примечание 2 — Криптонаборы, описанные в настоящих рекомендациях, соответствуют симметричной схеме защиты данных: векторы инициализации и ключи вычисления кода аутентификации сообщения и зашифрования сообщения, которые используются для формирования защищенной записи одной стороной, используются для проверки кода аутентификации сообщения и расшифрования сообщения другой стороной.

5 Протокол Record

Получив данные от протоколов более высокого уровня, протокол Record формирует из них последовательность структур, которые называются записями. Затем сформированные записи передаются транспортному протоколу.

Записи состоят из заголовка, описывающего передаваемые данные, и самих данных, которые передаются в открытом или защищенном виде в зависимости от текущего состояния соединения. Заголовок всегда передается в открытом виде.

Данные, полученные от протокола транспортного уровня, протокол Record интерпретирует как записи и формирует из них сообщения для протоколов верхнего уровня, опираясь на заголовки записей.

В настоящем разделе все действия будут описаны для одной фиксированной стороны взаимодействия (клиента или сервера), которая обладает ключом шифрования K_{ENC}^{write}, ключом вычисления кода аутентификации сообщения K_{MAC}^{write}, вектором инициализации IV^{write} для формирования записей и ключом расшифрования K_{ENC}^{read}, ключом проверки кода аутентификации сообщения K_{MAC}^{read}, вектором инициализации IV^{read} для чтения записей.

5.1 Состояние соединения

Состояние соединения определяет порядок обработки данных, передаваемых в рамках этого соединения. В каждый момент времени выделяется текущее состояние чтения и текущее состояние записи для каждой из сторон взаимодействия.

В каждый момент времени все записи обрабатываются в соответствии с текущим состоянием соединения. Установка и смена состояния соединения производятся после взаимного обмена сообщениями ChangeCipherSpec, производимого в рамках протокола Change Cipher Spec. Данное сообщение сигнализирует сторонам взаимодействия о том, что текущее состояние чтения/записи меняется на новое состояние, которое было согласовано в результате выполнения протокола Handshake.

Состояния чтения/записи характеризуются следующими значениями: параметрами сессии, параметрами соединения, номером получаемой/пересылаемой записи *seqnum* (числом от 0 до *SNMAX*, где параметр *SNMAX* задается выбранным криптонабором), который увеличивается на единицу после каждой полученной/отправленной записи. При смене состояния чтения/записи номеру пересылаемой записи должно присваиваться нулевое значение.

При установлении первого соединения алгоритм вычисления кода аутентификации и шифрования не используется, то есть все данные, передаваемые при работе протокола Handshake, будут пересылаться в открытом виде до тех пор, пока стороны не обменяются сообщениями ChangeCipherSpec и не сменят состояния чтения/записи, после чего все данные начнут передаваться в защищенном виде в соответствии с текущим состоянием чтения/записи.

При создании нового соединения в рамках текущего соединения все сообщения протокола Handshake передаются в защищенном виде в соответствии с состоянием чтения/записи, соответствующим текущему соединению, до тех пор, пока стороны не обменяются сообщениями ChangeCipherSpec и не сменят состояния чтения/записи.

5.2 Формирование записи

Каждая запись содержит следующие параметры: *type*, *version*, *length*, *fragment*. Первые три параметра *type*, *version*, *length* образуют заголовок записи, описанный в 5.2.2, и указывают на тип сообщения, версию протокола и длину передаваемых данных соответственно, а параметр *fragment* содержит фрагмент данных, передаваемых в открытом или защищенном виде. При этом с каждой записью неявно ассоциируется ее порядковый номер *seqnum*.

Часть данных, полученных от протоколов более высокого уровня и выделенных при фрагментации в соответствии с 5.2.1, переводится в TLSPlaintext-структуру, которая задается следующим образом:

```
struct {
     ContentType type;
     ProtocolVersion version = {3, 3}; /*TLS v1.2 */
     uint16 length;
     opaque fragment[TLSPlaintext.length];
} TLSPlaintext;
enum {
     change_cipher_spec(20), alert(21), handshake(22),
     application_data(23), (255)
} ContentType;
struct {
     uint8 major;
     uint8 minor;
} ProtocolVersion
```

Поле TLSPlaintext.fragment содержит фрагмент передаваемых данных в открытом виде.

При инициализации соединения до обмена сторонами сообщениями ChangeCipherSpec протокол Record оперирует незащищенными записями, которые формируются непосредственно в виде TLSPlaintext-структур.

После обмена сообщениями ChangeCipherSpec все данные пересылаются в защищенном виде. При установлении сторонами защищенного режима пересылки данных протокол Record переводит данные TLSPlaintext-структуры в защищенный вид, формируя из них TLSCiphertext-структуру (формат структуры полностью совпадает с форматом структуры TLSPlaintext), которой затем оперирует при информационном обмене:

```
struct {
     ContentType type;
     ProtocolVersion version = {3, 3}; /*TLS v1.2 */
     uint16 length;
     opaque fragment[TLSCiphertext.length];
} TLSCiphertext;
```

Поле TLSCiphertext.fragment содержит фрагмент передаваемых данных в защищенном виде и формируется из поля TLSPlaintext.fragment в соответствии с согласованным криптонабором. Процесс формирования этого поля описан в 5.2.3.

Формирование защищенной записи из фрагмента данных, выделенного при фрагментации в соответствии с 5.2.1, выполняется в соответствии со следующими этапами, схематично отраженными на рисунке 2:

1) формирование незащищенной записи, имеющей структуру TLSPlaintext;

2) вычисление кода аутентификации в соответствии с 5.2.3.2 от конкатенации номера записи, заголовка незащищенной записи и данных незащищенной записи;

3) зашифрование данных незащищенной записи и кода аутентификации, полученного на предыдущем шаге;

4) формирование защищенной записи, имеющей структуру TLSCiphertext.

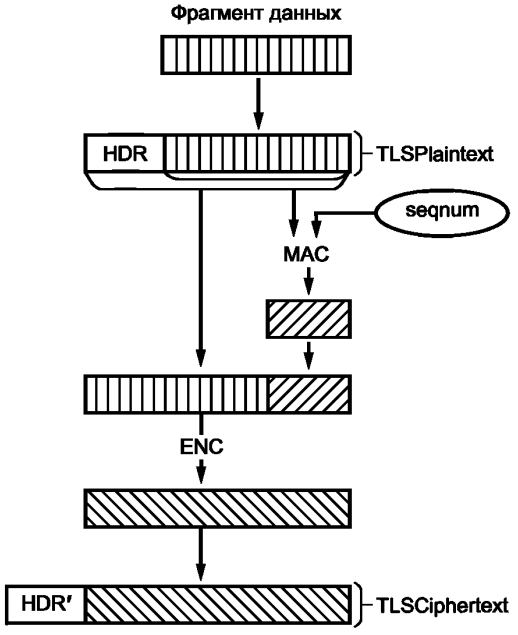


Рисунок 2 — Формирование защищенной записи

Примечание 1 — На рисунке 2 под HDR и HDR' подразумеваются заголовки незащищенной и защищенной записей соответственно.

Примечание 2 — В соответствии с [1] процесс формирования записи протокола TLS 1.2 допускает наличие этапа сжатия данных, однако при работе протокола в соответствии с криптонаборами, описанными в данных рекомендациях, этот этап отсутствует. В связи с этим структура TLSCompressed, описанная в [1], считается идентичной структуре TLSPlaintext и в тексте настоящих рекомендаций не используется.

5.2.1 Фрагментация

Принятые с верхнего уровня данные разбиваются протоколом Record на фрагменты, помещаемые в поля TLSPlaintext.fragment. Значение поля TLSPlaintext.length не должно превышать 2^{14} (то есть размер поля TLSPlaintext.fragment не должен превышать 16 Кбайт).

Если состояние соединения подразумевает защиту данных, то структура TLSPlaintext используется для формирования структуры TLSCiphertext. При этом размер поля fragment структуры TLSCiphertext может оказаться больше размера соответствующего поля структуры TLSPlaintext из-за добавления кода аутентификации сообщения (это приводит к тому, что значение поля TLSCiphertext.length становится больше значения поля TLSPlaintext.length). При работе протокола в соответствии с криптонаборами, описанными в данных рекомендациях, значение поля TLSCiphertext.length не должно превышать $2^{14} + 16$.

Одно сообщение протокола, работающего над протоколом Record, может быть разбито на несколько фрагментов. Одному фрагменту могут принадлежать данные нескольких сообщений одного типа, то есть сообщений одного протокола верхнего уровня. Запрещается посылать фрагменты нулевой длины для сообщений протокола Handshake и Change Cipher Spec. Запрещается фрагментировать сообщения протокола Alert.

Способы фрагментации не влияют на корректность работы всего протокола TLS 1.2 и должны определяться на этапе его реализации.

5.2.2 Формирование заголовка записи

Записи состоят из заголовка, описывающего передаваемые данные, и самого фрагмента данных. Каждая запись передается в открытом (TLSPlaintext-структура) или защищенном (TLSCiphertext-структура) виде в зависимости от текущего состояния соединения. В каждом из двух случаев заголовок передается в открытом виде, имеет длину 5 байт и состоит из следующих полей:

type — тип сообщения (1 байт). Всего определено четыре типа сообщений: сообщения протокола Change Cipher Spec (тип 20), сообщения протокола Alert (тип 21), сообщения протокола Handshake (тип 22), сообщения протокола Application Data (тип 23);

version — версия протокола (2 байта). Данный параметр содержит внутренний идентификатор версии протокола TLS. Внутренний идентификатор версии протокола TLS 1.2, описанного в настоящих рекомендациях, равен (3, 3);

length — длина фрагмента данных в байтах (2 байта). Данный параметр определяет количество байтов, передаваемых в записи после ее заголовка. Значения полей TLSPlaintext.length и TLSCiphertext.length должны удовлетворять ограничениям, определенным в 5.2.1.

5.2.3 Защита данных

В настоящем разделе предполагается, что незащищенная запись с номером *seqnum* обозначается далее через *Rec*:

```
TLSPlaintext Rec;
```

В настоящем разделе предполагается, что защищенная запись с номером *seqnum* обозначается далее через *PRec*:

```
TLSCiphertext PRec;
```

При обеспечении защиты данных с помощью вычисления кода аутентификации и шифрования значение поля *PRec.fragment* защищенной записи, соответствующей номеру *seqnum*, вычисляется следующим образом

$$PRec.fragment = RecENC, \tag{1}$$

где значение *RecENC* вычисляется согласно 5.2.3.3.

5.2.3.1 Алгоритм TLSTREE выработки ключей защиты записей

В настоящем разделе описывается алгоритм TLSTREE, используемый для порождения ключей защиты записей из корневого ключа $K_{root} \in B_{32}$.

$$TLSTREE(K_{root}, i) = Divers_3\left(Divers_2\left(Divers_1\left(K_{root}, STR_8\left(i \,\&\, C_1\right)\right), STR_8\left(i \,\&\, C_2\right)\right), STR_8\left(i \,\&\, C_3\right)\right), \tag{2}$$

где

- число $i \in \{0,1,\ldots,2^{64}-1\}$;
- константы C_1, C_2, $C_3 \in \mathbb{Z}_{2^{64}}$ определяются конкретным криптонабором;
- $Divers_j(K,D)$, $j \in \{1,2,3\}$ — алгоритм диверсификации ключа $K \in B_{32}$ по данным диверсификации $D \in B_8$, который задается с помощью функции KDF_{256}, определяемой алгоритмом KDF_GOSTR3411_2012_256, описанным в Р 50.1.113—2016 (подраздел 4.4):

$$\begin{aligned} Divers_1(K,D) &= KDF_{256}(K,"level1",D);\\ Divers_2(K,D) &= KDF_{256}(K,"level2",D);\\ Divers_3(K,D) &= KDF_{256}(K,"level3",D). \end{aligned} \tag{3}$$

5.2.3.2 Код аутентификации сообщения

В настоящем разделе предполагается, что ключ K_{MAC} соответствует ключу K_{MAC}^{write} для состояния записи и ключу K_{MAC}^{read} для состояния чтения.

Для записи *Rec* с номером *seqnum* код аутентификации вычисляется от следующих данных:

$$MACData = STR_8(seqnum) \,|\, Rec.type \,|\, Rec.version \,|\, Rec.length \,|\, Rec.fragment. \tag{4}$$

Значение ключа K_{MAC}^{seqnum}, соответствующего конкретному номеру *seqnum*, получается из ключа K_{MAC} с помощью алгоритма TLSTREE, описанного в 5.2.3.1, в соответствии с формулой

$$K_{MAC}^{seqnum} = TLSTREE(K_{MAC}, seqnum). \tag{5}$$

Параметры алгоритма вычисления кода аутентификации сообщения *MAC* определяются согласованными криптонаборами в соответствии с 10.3.

Для записи *Rec* с номером *seqnum* код аутентификации сообщения *RecMAC* определяется в соответствии с формулой

$$RecMAC = MAC\left(K_{MAC}^{seqnum}, MACData\right). \tag{6}$$

5.2.3.3 Шифрование

В настоящих рекомендациях описываются криптонаборы, для которых использование алгоритма шифрования при формировании защищенной записи является обязательным.

В настоящем разделе предполагается, что ключ K_{ENC} соответствует ключу K_{ENC}^{write} для состояния записи и ключу K_{ENC}^{read} для состояния чтения. Также полагаем, что вектор инициализации IV соответствует вектору инициализации IV^{write} для состояния записи и вектору инициализации IV^{read} для состояния чтения.

Для записи Rec с номером $seqnum$ зашифровываются следующие данные:

$$ENCData = Rec.fragment \mid RecMAC. \tag{7}$$

Значение ключа K_{ENC}^{seqnum}, соответствующее конкретному номеру записи $seqnum$, получается из ключа K_{ENC} с помощью алгоритма TLSTREE, описанного в 5.2.3.1, в соответствии с формулой

$$K_{ENC}^{seqnum} = TLSTREE\left(K_{ENC}, seqnum\right). \tag{8}$$

Значение вектора инициализации IV_{seqnum}, соответствующее конкретному номеру записи $seqnum$, определяется следующим образом

$$IV_{seqnum} = STR_{n/2}\left(\left(INT\left(IV\right) + seqnum\right) \bmod 2^{n \cdot 8/2}\right). \tag{9}$$

Параметры алгоритма шифрования ENC определяются согласованным криптонабором. Значение $RecENC$ вычисляется в соответствии с формулой

$$\text{Rec}ENC = ENC\left(K_{ENC}^{seqnum}, IV_{seqnum}, ENCData\right), \tag{10}$$

при этом зашифрование данных проводится в режиме CTR-ACPKM, описанном в Р 1323565.1.017—2018 и использующем блочный шифр, соответствующий согласованному криптонабору.

6 Протокол Handshake

Протокол Handshake работает поверх протокола Record и используется для согласования следующих параметров сессии:

идентификатор сессии — произвольная последовательность байтов, выработанная сервером для идентификации активной или возобновляемой сессии;

сертификат стороны — сертификат в формате X.509, формирующийся в соответствии с [2]. Аутентификация опционально может быть односторонней (аутентификация сервера клиентом, сертификат предоставляет только сервер) или взаимной (взаимная аутентификация сервера и клиента, клиент и сервер обмениваются сертификатами);

идентификатор криптонабора — идентификатор, определяющий набор алгоритмов и их параметров, использующихся в рамках данной сессии;

значение *MS* (master secret) — общее секретное значение длиной 48 байт, которое вырабатывается сторонами с помощью протокола выработки и подтверждения ключа (см. 6.4);

значение флага возобновления — флаг, разрешающий/запрещающий повторные соединения в рамках данной сессии.

Протокол Handshake начинает свою работу с сообщения ClientHello, которое может быть послано клиентом по его собственной инициативе либо в качестве ответа на сообщение сервера HelloRequest.

Схема обмена сообщениями в протоколе Handshake может быть полной или упрощенной.

Полная схема обмена сообщениями (см. рисунок 3) используется, если клиент в сообщении приветствия ClientHello в качестве идентификатора сессии указал пустую строку либо если сервер не смог найти в кэше сессий параметры сессии, соответствующей данному идентификатору.

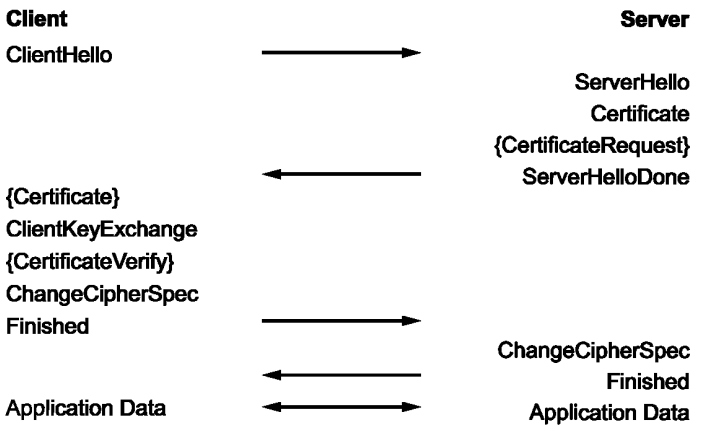


Рисунок 3 — Полная схема обмена сообщениями в протоколе Handshake

Примечание 1 — Фигурными скобками отмечены те сообщения, которые являются опциональными.

Примечание 2 — Сообщение ChangeCipherSpec формально относится не к протоколу Handshake, а к протоколу Change Cipher Spec.

Примечание 3 — В соответствии с [1] протокол TLS 1.2 допускает возможность установления анонимного соединения без обмена сертификатами, однако в протоколе, соответствующем криптонаборам, описанным в данных рекомендациях, данная опция использоваться не должна.

Если клиент хочет создать соединение в рамках существующей сессии, то в сообщении приветствия ClientHello он должен указать идентификатор этой сессии. В случае если сервер смог найти в кэше сессий параметры сессии, соответствующей данному идентификатору, процесс обмена сообщениями протокола Handshake соответствует упрощенной схеме (см. рисунок 4).

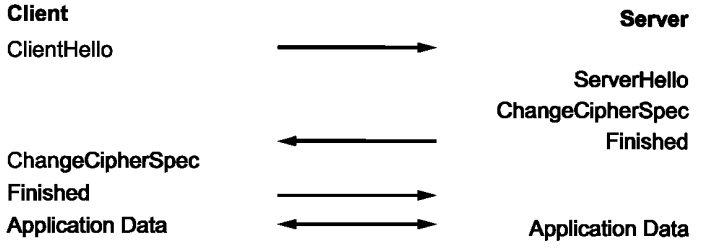


Рисунок 4 — Упрощенная схема обмена сообщениями в протоколе Handshake

Во время упрощенной схемы работы протокола Handshake стороны не вырабатывают новое общее сессионное секретное значение *MS*. Для выработки ключевого материала, необходимого для работы протокола Record в рамках устанавливаемого соединения, используется значение *MS*, выработанное при выполнении полной схемы протокола Handshake, в ходе которой были установлены параметры сессии с данным идентификатором.

6.1 Схема взаимодействия сторон

В таблице 6.1.1 пунктирной линией обозначены все опциональные сообщения, которые передаются в случае двухсторонней аутентификации.

Таблица 6.1.1 — Схема взаимодействия сторон в протоколе Handshake

Формирование параметров и действия со стороны клиента *C*	Передаваемые сообщения	Формирование параметров и действия со стороны сервера *S*
Выработка случайного значения r_C	**ClientHello** r_C →	

Продолжение таблицы 6.1.1

Формирование параметров и действия со стороны клиента C	Передаваемые сообщения	Формирование параметров и действия со стороны сервера S
	ServerHello ← r_S	Выработка случайного значения r_S
Этап выработки общего секрета		
	Certificate ← $Cert_S$ (содержит ключ Q_S)	
	CertificateRequest ←	
	ServerHelloDone ←	
	Certificate → $Cert_C$ (содержит ключ Q_C)	
Выработка общего секрета PS, (выбирается случайно из B_{32})		
Формирование эфемерного ключа на кривой, соответствующей Q_S: $Q_{EPH} = k_{EPH}P$, где k_{EPH} выбирается случайно из $\mathbb{Z}_q^*$		
$H = HASH(r_C \mid r_S)$; $K_{MAC}^{Exp} \mid K_{ENC}^{Exp} = KEG(k_{EPH}, Q_S, H)$		
Формирование экспортного представления общего секрета PS: $IV = H[25..24 + n/2]$; $PSExp = KExp15(PS, K_{MAC}^{Exp}, K_{ENC}^{Exp}, IV)$	**ClientKeyExchange** → Q_{EPH}, $PSExp$	$H = HASH(r_C \mid r_S)$; $K_{MAC}^{Exp} \mid K_{ENC}^{Exp} = KEG(k_S, Q_{EPH}, H)$
		Извлечение общего секрета из экспортного представления: $IV = H[25..24 + n/2]$; $PS = KImp15(PSExp, K_{MAC}^{Exp}, K_{ENC}^{Exp}, IV)$
$sgn_C = SIGN_{k_C}(HM)$	**CertificateVerify** → sgn_C	Проверка sgn_C
Этап выработки ключей из значения PS		
Выработка общего сессионного секретного значения: $MS = PRF(PS, "extended\ master\ secret", HASH(HM))$;		Выработка общего сессионного секретного значения: $MS = PRF(PS, "extended\ master\ secret", HASH(HM))$;
Генерация ключевого материала соединения: $K_{MAC,C}^{write} \mid K_{MAC,C}^{read} \mid K_{ENC,C}^{write} \mid K_{ENC,C}^{read} \mid IV_C^{write} \mid IV_C^{read} = PRF(MS, "key\ expansion", r_S \mid r_C)$		Генерация ключевого материала соединения: $K_{MAC,S}^{read} \mid K_{MAC,S}^{write} \mid K_{ENC,S}^{read} \mid K_{ENC,S}^{write} \mid IV_S^{read} \mid IV_S^{write} = PRF(MS, "key\ expansion", r_S \mid r_C)$

Окончание таблицы 6.1.1

Формирование параметров и действия со стороны клиента *C*	Передаваемые сообщения	Формирование параметров и действия со стороны сервера *S*
Этап подтверждения ключа		
	ChangeCipherSpec →	
client _verify _data = $PRF(MS, "client\ finished", HASH(HM))$;	**Finished** → *client_verify_data*	
	ChangeCipherSpec ←	
	Finished ← *server_verify_data*	*server _verify _data* = $PRF(MS, "server\ finished", HASH(HM))$;

6.2 Формат сообщений протокола Handshake

Каждое сообщение протокола Handshake содержит специальный заголовок, состоящий из четырех байтов. Первый байт содержит код типа сообщения, три следующих байта — длину сообщения:

```
enum {
    hello_request(0), client_hello(1), server_hello(2),
    certificate(11), certificate_request(13),
    server_hello_done(14), certificate_verify(15),
    client_key_exchange(16), finished(20), (255)
} HandshakeType;
struct {
    HandshakeType msg_type;    /* handshake type */
    uint24 length;             /* bytes in message */
    select (HandshakeType) {
        case hello_request:       HelloRequest;
        case client_hello:        ClientHello;
        case server_hello:        ServerHello;
        case certificate:         Certificate;
        case certificate_request: CertificateRequest;
        case server_hello_done:   ServerHelloDone;
        case certificate_verify:  CertificateVerify;
        case client_key_exchange: ClientKeyExchange;
        case finished:            Finished;
    } body;
} Handshake;
```

Сообщения протокола Handshake, описанные ниже, приведены в том порядке, в котором они должны следовать для обеспечения корректной работы протокола. Единственным исключением является сообщение запроса приветствия (HelloRequest), которое может быть послано сервером в любой момент, но которое должно быть проигнорировано клиентом, если оно получено во время выполнения протокола Handshake.

6.3 Сообщения приветствия

6.3.1 Сообщение запроса приветствия (HelloRequest)

Данное сообщение может быть послано сервером в любой момент и используется для оповещения клиента о том, что тот должен начать процесс согласования параметров сессии заново. В ответ на данный запрос клиент должен послать сообщение приветствия ClientHello.

Данное сообщение должно быть проигнорировано клиентом, если в этот момент он уже участвует в процессе согласования параметров сессии. В случае если клиент отказывается пересогласовывать параметры сессии, то он может проигнорировать это сообщение либо послать в ответ оповещение no_renegotiation (см. раздел 8).

Если сервер послал сообщение запроса приветствия HelloRequest, но не дождался сообщения ClientHello в качестве ответа со стороны клиента, он может закрыть соединение, передав оповещение об ошибке с уровнем fatal (см. раздел 8).

После отправки сообщения запроса приветствия HelloRequest сервер не должен повторно отправлять данное сообщение до тех пор, пока последующий процесс согласования параметров сессии на этапе выполнения протокола Handshake не будет завершен.

Данное сообщение не должно входить в переменную *HM*.

6.3.2 Сообщение приветствия клиента (ClientHello)

Клиент посылает сообщение приветствия ClientHello в одном из следующих случаев:

- клиент впервые подключается к серверу;
- как ответ на сообщение запроса приветствия HelloRequest;
- с целью пересогласования параметров сессии.

Сообщение ClientHello содержит следующие параметры:

версия протокола (client_version) — максимальная версия протокола из тех, которые готов поддерживать клиент [настоящие рекомендации описывают криптонаборы для протокола TLS 1.2, для которого значение версии равняется (3, 3)];

строка со случайными данными (random) — строка данных длины 32 байта, выработанная клиентом, в которой первые 4 байта занимает значение текущего времени в 32-битном формате UNIX [количество секунд, прошедших с полуночи (00:00:00 UTC) 1 января 1970 года], а остальные 28 байт занимает строка, выработанная случайным образом;

идентификатор сессии (session_id) — идентификатор сессии, выбранный клиентом. Если данное поле пусто, значит клиент хочет согласовать параметры новой сессии;

список криптонаборов (cipher_suites) — список криптонаборов, которые поддерживает клиент. Порядок криптонаборов в списке отражает их степень предпочтения (предпочтительные идут первыми). Если поле ClientHello.session_id не пустое, поле ClientHello.cipher_suites должно содержать идентификатор криптонабора, согласованного во время установления параметров сессии, имеющей идентификатор ClientHello.session_id;

список методов сжатия (compression_methods) — данная версия протокола не поддерживает методы сжатия, и поле должно содержать один метод с идентификатором null (0);

расширения (extensions) — криптонаборы, описанные в настоящем документе, предполагают наличие трех обязательных расширений со стороны клиента: signature_algorithms, extended_master_secret и renegotiation_info.

Сообщение ClientHello имеет следующую структуру:

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<2..2^16-2>;
    CompressionMethod compression_methods<1..2^8-1>;
    Extension extensions<0..2^16-1>;
} ClientHello;
```

После отправки сообщения ClientHello клиент ожидает от сервера сообщения ServerHello. В ответ на любое другое сообщение протокола Handshake (за исключением сообщения запроса приветствия HelloRequest) клиент должен ответить оповещением об ошибке с уровнем fatal (см. раздел 8).

6.3.3 Сообщение приветствия сервера (ServerHello)

Данное сообщение отправляется сервером после получения им сообщения ClientHello при условии, что среди параметров, переданных клиентом в сообщении приветствия, присутствует поддерживаемый сервером набор параметров установления соединения.

Сообщение приветствия содержит следующие параметры:

версия протокола (server_version) — максимальная поддерживаемая сервером версия протокола, не превосходящая версии клиента client_version.

Примечание — Организация обратной совместимости проводится в соответствии с приложением Е документа [1];

строка со случайными данными (random) — строка данных длины 32 байта, выработанная сервером, в которой первые 4 байта занимает значение текущего времени в 32-битном формате UNIX [количество секунд, прошедших с полуночи (00:00:00 UTC) 1 января 1970 года], а остальные 28 байт занимает строка, выработанная случайным образом;

идентификатор сессии (session_id) — идентификатор сессии, в рамках которой осуществляется соединение. Если поле ClientHello.session_id не было пустым, тогда в кэше сессий сервер ищет параметры сессии, соответствующей данному идентификатору. В случае если такая сессия существует и сервер готов создать соединение в рамках данной сессии, он присваивает то же значение идентификатора сессии полю ServerHello.session_id. Если поле ClientHello.session_id было пустым либо если сервер не нашел параметры сессии, соответствующей данному идентификатору, сервер задает новое значение идентификатора, указывающего на создаваемую сессию. Сервер может послать пустое значение идентификатора, это будет означать, что параметры данной сессии не будут храниться в кэше сервера и повторное создание соединения в рамках данной сессии будет невозможно;

криптонабор (cipher_suite) — криптонабор, который сервер выбирает из списка ClientHello.cipher_suites. Если значение ClientHello.session_id не нулевое, данный параметр должен содержать идентификатор криптонабора, согласованного в сессии ClientHello.session_id;

метод сжатия (compression_method) — данная версия протокола не поддерживает методы сжатия, и поле должно содержать один метод с идентификатором null (0);

расширения (extensions) — криптонаборы, описанные в настоящем документе, предполагают наличие двух обязательных расширений со стороны сервера: extended_master_secret и renegotiation_info.

Сообщение ServerHello имеет следующую структуру:

```
struct {
    ProtocolVersion server_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
    Extension extensions<0..2^16-1>;
} ServerHello;
```

6.3.4 Расширения (extensions)

Каждое расширение, посылаемое клиентом и сервером, имеет следующую структуру:

```
struct {
    ExtensionType extension_type;
    opaque extension_data<0..2^16-1>;
} Extension;
enum {
    signature_algorithms(13),
    extended_master_secret(23),
    renegotiation_info(65281),(65535)
} ExtensionType;
```

6.3.4.1 Расширение signature_algorithms

Расширение signature_algorithms является обязательным только для клиента и не пересылается сервером. Данное расширение содержит список пар алгоритмов подписи и хэширования, поддерживаемых клиентом.

Данное расширение имеет тип 13. Поле extension_data расширения signature_algorithms, пересылаемого клиентом, содержит параметр supported_signature_algorithms, который задается следующим способом:

```
SignatureAndHashAlgorithm
    supported_signature_algorithms<2..2^16-2>;
struct {
    HashAlgorithm hash;
    SignatureAlgorithm signature;
} SignatureAndHashAlgorithm;
```

Множество поддерживаемых алгоритмов хэширования задается следующим образом:

```
enum {
    gostr34112012_256(238),
    gostr34112012_512(239), (255)
} HashAlgorithm;
```

При этом значения gostr34112012_256 и gostr34112012_512 соответствуют алгоритму хэширования, определенному в ГОСТ Р 34.11—2012, с длиной выхода 256 и 512 бит соответственно.

Множество поддерживаемых алгоритмов подписи задается следующим образом:

```
enum {
    gostr34102012_256(238),
    gostr34102012_512(239), (255)
} SignatureAlgorithm;
```

При этом значения gostr34102012_256 и gostr34102012_512 соответствуют алгоритму подписи, определенному в ГОСТ Р 34.10—2012, с длиной ключа 256 и 512 бит соответственно.

В протоколе, соответствующем криптонаборам, описанным в данных рекомендациях, допускается использование следующих пар алгоритмов подписи и хэширования: (gostr34112012_256, gostr34102012_256) и (gostr34112012_512, gostr34102012_512).

Примечание — В рамках протокола, соответствующего криптонаборам, описанным в данных рекомендациях, указание поддерживаемого алгоритма хэширования является избыточным и используется для обеспечения совместимости с форматом расширения signature_algorithms, описанным в [1].

6.3.4.2 Расширение extended_master_secret

Данное расширение всегда посылается клиентом и сервером и сигнализирует о том, что клиент и сервер используют расширенный вариант генерации значения общего секретного значения *MS* в соответствии с 6.4.7.

Данное расширение пересылается пустым и имеет тип 23.

Примечание — Описание расширения extended_master_secret соответствует [3].

6.3.4.3 Расширение renegotiation_info

Данное расширение всегда посылается клиентом и сервером и связывает каждое новое повторное согласование соединения с предшествующими сообщениями Finished, что препятствует внедрению сообщений противника.

Данное расширение имеет тип 65281. Поле extension_data расширения renegotiation_info, пересылаемого клиентом, содержит структуру RenegotiationInfo, которая задается следующим образом:

```
struct {
    opaque renegotiated_connection<0..255>;
} RenegotiationInfo;
```

В случае если сообщения протокола Handshake посылаются впервые, то есть не в рамках ранее созданного соединения, поле renegotiated_connection должно оставаться пустым как в сообщении ClientHello, так и в сообщении ServerHello.

В случае если сообщение приветствия клиента ClientHello передается в рамках ранее созданного соединения, поле renegotiated_connection должно содержать данные client_verify_data, которые пересылались клиентом в поле verify_data сообщения Finished во время выполнения протокола Handshake, в результате которого было создано текущее соединение.

В случае если сообщение приветствия клиента ServerHello было передано в результате повторного согласования соединения, поле renegotiated_connection должно содержать конкатенацию данных client_verify_data и server_verify_data, которые пересылались клиентом и сервером соответственно в поле verify_data сообщения Finished во время выполнения протокола Handshake, в результате которого было создано текущее соединение.

В случае отсутствия корректно сформированного расширения renegotiation_info в сообщениях ClientHello или ServerHello проверяющая данное сообщение сторона должна ответить оповещением об ошибке с уровнем fatal (см. раздел 8).

Примечание — Описание расширения renegotiation_info соответствует [4].

6.4 Выработка и подтверждение ключа

6.4.1 Сообщение с сертификатом сервера (Certificate)

Данное сообщение посылается сервером и содержит цепочку сертификатов. Сертификат сервера следует первым в данной цепочке и содержит открытый ключ сервера.

Каждый сертификат из цепочки сертификатов сервера должен быть подписан в соответствии с одной из пар алгоритмов из расширения signature_algorithms. Открытый ключ сервера должен соответствовать одной из пар алгоритмов подписи и хэширования, указанной в расширении signature_algorithms. При этом алгоритм подписи, использовавшийся для подписания сертификатов из цепочки сертификатов сервера, может отличаться от алгоритма подписи, соответствующего хранящемуся в сертификате сервера открытому ключу сервера.

Структура данного сообщения выглядит следующим образом:

```
opaque ASN.1Cert<1..2^24-1>;

struct {
    ASN.1Cert certificate_list<0..2^24-1>;
} Certificate;
```

6.4.2 Сообщение запроса сертификата (CertificateRequest)

Данное сообщение посылается сервером в случае взаимной аутентификации.

Сообщение запроса сертификата содержит следующие параметры:

типы сертификатов (certificate_types) — настоящие рекомендации определяют два типа сертификатов: gostr34102012_256, gostr34102012_512. Данные типы указывают на то, что сервер принимает сертификаты клиента с ключами проверки подписи, соответствующими алгоритму ГОСТ Р 34.10—2012 с длиной ключа 256 и 512 бит соответственно;

поддерживаемые алгоритмы подписи и хэширования (supported_signature_algorithms) — пары алгоритмов подписи и хэширования, указывающие на то, с помощью каких алгоритмов могут быть подписаны сертификаты из цепочки сертификатов клиента. Ключ проверки подписи, хранящийся в сертификате клиента, должен также соответствовать одному из алгоритмов подписи (не обязательно тому же, которым подписан сертификат), перечисленных в поле supported_signature_algorithms;

удостоверяющие центры (certificate_authorities) — список допустимых удостоверяющих центров (УЦ), представленных в формате кодирования DER. Данный список может задавать допустимые имена для корневых УЦ или для подчиненных УЦ. Если данное поле остается пустым, то клиент может посылать любой сертификат, соответствующий полю certificate_types.

Структура данного сообщения выглядит следующим образом:

```
struct {
    ClientCertificateType certificate_types<1..2^8-1>;
    SignatureAndHashAlgorithm
       supported_signature_algorithms<2..2^16-2>;
    DistinguishedName certificate_authorities<0..2^16-1>;
} CertificateRequest;
```

Множество типов сертификатов задается следующим образом:

```
enum {
    gostr34102012_256(238),
    gostr34102012_512(239), (255)
} ClientCertificateType;
```

Структура SignatureAndHashAlgorithm задается в 6.3.4.1.
Тип DistinguishedName задается следующим образом:

```
opaque DistinguishedName<1..2^16-1>;
```

6.4.3 Сообщение о завершении приветствия (ServerHelloDone)

Данное сообщение посылается сервером и сигнализирует о том, что все сообщения этапа приветствия переданы и сервер переходит в состояние ожидания сообщений от клиента.

Структура данного сообщения выглядит следующим образом:

```
struct { } ServerHelloDone;
```

6.4.4 Сообщение с сертификатом клиента (Certificate)

Данное сообщение посылается клиентом в случае двухсторонней аутентификации, то есть как ответ на сообщение о запросе сертификата со стороны сервера CertificateRequest. Данное сообщение содержит цепочку сертификатов. Сертификат клиента следует первым в данной цепочке и содержит ключ проверки подписи клиента.

Каждый из сертификатов цепочки сертификата клиента должен быть подписан в соответствии с одной из пар алгоритмов из списка CertificateRequest.supported_signature_algorithms.

Ключ проверки подписи в сертификате клиента должен соответствовать алгоритму подписи, указанному в расширении CertificateRequest.certificate_types.

Структура данного сообщения идентична структуре в 6.4.1.

6.4.5 Сообщение с ключом обмена клиента (ClientKeyExchange)

Для формирования сообщения ClientKeyExchange клиент выполняет следующие шаги:

- вырабатывает секретное значение PS, которое выбирается случайно из множества B_{32};
- выбирает случайное значение k_{EPH} из множества $\mathbb{Z}_q^*$;
- формирует эфемерный ключ $Q_{EPH} = k_{EPH} \cdot P$;
- с помощью алгоритма KEG, описанного в 6.4.5.1, формирует ключи экспорта K_{MAC}^{Exp} и K_{ENC}^{Exp} следующим образом

$$H = HASH(r_C \mid r_S);$$
$$K_{MAC}^{Exp} \mid K_{ENC}^{Exp} = KEG(k_{EPH}, Q_S, H); \quad (11)$$

- формирует экспортное представление общего секрета PS:

$$IV = H[25..24 + n/2];$$
$$PSExp = KExp15(PS, K_{MAC}^{Exp}, K_{ENC}^{Exp}, IV). \quad (12)$$

Структура данного сообщения выглядит следующим образом:

```
enum { vko_kdf_gost } KeyExchangeAlgorithm;

struct {
    select (KeyExchangeAlgorithm) {
```

```
        case vko_kdf_gost: PSKeyTransport;
    } exchange_keys;
} ClientKeyExchange;
```

Структура PSKeyTransport определяется следующим образом:

```
PSKeyTransport ::= SEQUENCE {
    PSExp                   OCTET STRING,
    ephemeralPublicKey      SubjectPublicKeyInfo
}

SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm               AlgorithmIdentifier,
    subjectPublicKey        BITSTRING
}

AlgorithmIdentifier ::= SEQUENCE {
    algorithm               OBJECT IDENTIFIER,
    parameters              ANY OPTIONAL
}
```

Здесь в поле PSExp передается экспортное представление ключа *PSExp*, а поле ephemeralPublicKey задается структурой SubjectPublicKeyInfo (см. ГОСТ Р ИСО/МЭК 9594-8, раздел 8) и содержит информацию о параметре Q_{EPH}.

Получив сообщение ClientKeyExchange, сервер выполняет следующие шаги:

- проверяет выполнение следующих трех условий и в случае невыполнения одного из них прекращает взаимодействие, посылая сообщение об ошибке с уровнем fatal (см. раздел 8):

1) точка Q_{EPH} принадлежит кривой, соответствующей открытому ключу сервера;

2) точка Q_{EPH} не равна нулевой точке O;

3) $q \cdot Q_{EPH} = O$;

- формирует ключи экспорта K_{MAC}^{Exp} и K_{ENC}^{Exp}:

$$H = HASH(r_C \mid r_S);$$
$$K_{MAC}^{Exp} \mid K_{ENC}^{Exp} = KEG(k_S, Q_{EPH}, H); \tag{13}$$

- извлекает общий секрет PS из экспортного представления *PSExp*:

$$IV = H[25..24 + n/2];$$
$$PS = KImp15(PSExp, K_{MAC}^{Exp}, K_{ENC}^{Exp}, IV). \tag{14}$$

6.4.5.1 Алгоритм выработки ключей экспорта KEG

Алгоритм выработки ключей экспорта KEG принимает на вход закрытый ключ d, открытый ключ Q и строку $h \in B_{32}$. Результатом работы данного алгоритма является строка длины 64 байта.

Алгоритм выработки ключей экспорта задается двумя способами в зависимости от длины закрытого ключа.

В случае если длина закрытого ключа d равна 64 байта (512 бит), алгоритм KEG определяется следующим образом

$$KEG(d, Q, h) = VKO_{512}(d, Q, UKM), \tag{15}$$

где в качестве функции VKO_{512} используется алгоритм VKO_GOSTR3410_2012_512, определенный в Р 50.1.113—2016, а параметр UKM определяется следующим образом

$$UKM = \begin{cases} r, \text{если } r \neq 0 \\ 1, \text{если } r = 0 \end{cases}, \text{ где } r = INT(h[1..16]). \tag{16}$$

В случае если длина закрытого ключа d равна 32 байта (256 бит), алгоритм KEG определяется следующим образом:

$$KEG(d,Q,h) = KDFTREE_{256}(K_{EXP},"kdf\ tree",seed,1), \tag{17}$$

где в качестве функции $KDFTREE_{256}$ используется алгоритм диверсификации KDF_TREE_GOSTR3411_2012_256, определенный в Р 50.1.113—2016, результатом работы которого является строка $K(1)|K(2) \in B_{64}$, а параметры K_{EXP} и $seed$ задаются следующим образом:

$$K_{EXP} = VKO_{256}(d,Q,UKM); \tag{18}$$

$$UKM = \begin{cases} r, \text{если } r \neq 0 \\ 1, \text{если } r = 0 \end{cases};$$

$$r = INT(h[1..16]);$$

$$seed = h[17..24].$$

где в качестве функции VKO_{256} используется алгоритм VKO_GOSTR3410_2012_256, определенный в Р 50.1.113—2016.

6.4.6 Сообщение проверки сертификата (CertificateVerify)

В случае если сервером было послано сообщение запроса сертификата CertificateRequest, клиентом пересылается сообщение проверки сертификата CertificateVerify, которое содержит в себе значение

$$sgn_C = SIGN_{k_C}(HM) = str_l(r)|str_l(s), \tag{19}$$

где

- $SIGN_{k_C}$ — алгоритм подписи ГОСТ Р 34.10—2012 на ключе подписи клиента k_C, соответствующем ключу проверки подписи Q_C из сертификата клиента, результатом работы которого являются два числа r и s, определенных в ГОСТ Р 34.10—2012;
- для сертификатов, соответствующих алгоритму подписи gostr34102012_256, $l = 32$, а для сертификатов, соответствующих алгоритму gostr34102012_512, $l = 64$.

Примечание — Переменная HM в данном случае является конкатенацией всех сообщений протокола Handshake, начиная с сообщения приветствия ClientHello и заканчивая сообщением CertificateVerify (не включая данное сообщение).

Структура данного сообщения выглядит следующим образом:

```
struct {
    SignatureAndHashAlgorithm algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

При этом поле CertificateVerify.signature содержит значение sgn_C.

Получив сообщение CertificateVerify, сервер проверяет значение подписи с помощью ключа проверки подписи Q_C, полученного из сертификата клиента.

6.4.7 Сообщение завершения протокола Handshake (Finished)

Клиент и сервер вырабатывают общее секретное значение MS длины 48 байт:

$$MS = LMB_{48}(PRF(PMS,"extended\ master\ secret",HASH(HM))). \tag{20}$$

Примечание — Переменная HM в данном случае является конкатенацией всех сообщений протокола Handshake, начиная с сообщения приветствия ClientHello и заканчивая сообщением Finished со стороны клиента (не включая данное сообщение).

Клиент вырабатывает ключи вычисления кода аутентификации, ключи шифрования и векторы инициализации для каждого из состояний чтения/записи:

$$K^{write}_{MAC,C}|K^{read}_{MAC,C}|K^{write}_{ENC,C}|K^{read}_{ENC,C}|IV^{write}_{C}|IV^{read}_{C} = LMB_{4k+n}(PRF(MS,"key\ expansion",r_S|r_C)). \tag{21}$$

Сервер вырабатывает ключи вычисления кода аутентификации, ключи шифрования и векторы инициализации для каждого из состояний чтения/записи:

$$K^{read}_{MAC,S} \mid K^{write}_{MAC,S} \mid K^{read}_{ENC,S} \mid K^{write}_{ENC,S} \mid IV^{read}_{S} \mid IV^{write}_{S} = LMB_{4k+n}\left(PRF\left(MS,"key\ expansion",r_S \mid r_C\right)\right). \quad (22)$$

Сразу после отправки сообщения изменения состояния ChangeCipherSpec клиент посылает сообщение Finished, которое содержит в себе данные client_verify_data, сформированные следующим образом

$$client_verify_data = PRF\left(MS,"client\ finished",HASH(HM)\right). \quad (23)$$

При получении сообщения Finished, переданного со стороны клиента, сервер посылает свою пару сообщений ChangeCipherSpec и Finished. При этом данные server_verify_data, пересылаемые в сообщении Finished, формируются в соотвествии с формулой

$$server_verify_data = PRF\left(MS,"server\ finished",HASH(HM)\right). \quad (24)$$

Примечание — Переменная *HM* в данном случае является конкатенацией всех сообщений протокола Handshake, начиная с сообщения приветствия ClientHello и заканчивая сообщением Finished со стороны клиента (не включая данное сообщение) в случае формирования client_verify_data либо сообщением Finished со стороны сервера (не включая данное сообщение) в случае формирования server_verify_data. Таким образом, значение переменной *HM* при формировании сообщения Finished со стороны клиента будет отличаться от значения переменной *HM* при формировании сообщения Finished со стороны сервера.

Структура данного сообщения выглядит следующим образом:

```
struct {
    opaque verify_data[verify_data_length];
} Finished;
```

где параметр verify_data_length принимает значение 32.

При получении сообщения Finished клиент и сервер должны проверить корректность данных, пересылаемых в поле verify_data.

7 Протокол Change Cipher Spec

Сообщение протокола Change Cipher Spec, пересылаемое обоими участниками протокола, сигнализирует о том, что с данного момента сторона, пославшая данное сообщение, устанавливает защищенный режим пересылки данных, который соответствует новому состоянию соединения, параметры которого были выработаны в ходе работы протокола Handshake.

Сообщение протокола Change Cipher Spec состоит из одного байта, принимающего значение 1.

```
struct {
    enum { change_cipher_spec(1), (255) } type;
} ChangeCipherSpec;
```

8 Протокол Alert

Каждое сообщение протокола Alert содержит информацию о пересылаемом оповещении:

```
struct {
    AlertLevel level;
    AlertDescription description;
} Alert;
```

Поле level (уровень оповещения) задается одним байтом и может принимать значения warning и fatal:

- Оповещения с уровнем warning не требуют немедленного прекращения соединения в общем случае. При получении данного оповещения сторона взаимодействия может принять решение

о продолжении работы или о закрытии соединения. В последнем случае она должна послать ответное оповещение с уровнем fatal.

- Оповещения с уровнем fatal должны приводить к немедленному прекращению соединения. В этом случае другие соединения, относящиеся к данной сессии, могут продолжать функционировать, однако идентификатор сессии обязан быть аннулирован (то есть исключен из кэша ранее созданных сессий), чтобы предотвратить возможность создания нового соединения, использующего параметры данной сессии.

```
enum {
    warning(1),
    fatal(2), (255)
} AlertLevel;
```

Поле description (описание оповещения) задается одним байтом, соответствующим определенному типу оповещения, описанному в структуре AlertDescription:

```
enum {
    close_notify(0),
    unexpected_message(10),
    bad_record_mac(20),
    record_overflow(22),
    handshake_failure(40),
    bad_certificate(42),
    unsupported_certificate(43),
    certificate_revoked(44),
    certificate_expired(45),
    certificate_unknown(46),
    illegal_parameter(47),
    unknown_ca(48),
    access_denied(49),
    decode_error(50),
    decrypt_error(51),
    protocol_version(70),
    insufficient_security(71),
    internal_error(80),
    user_canceled(90),
    no_renegotiation(100),
    unsupported_extension(110), (255)
} AlertDescription;
```

Примечание — В протоколе, соответствующем криптонаборам, описанным в данных рекомендациях, оповещения decryption_failed_RESERVED(21), decompression_failure(30), no_certificate_RESERVED(41) и export_restriction_RESERVED(60), описанные в [1], пересылаться не должны.

Оповещения посылаются сторонами взаимодействия в двух случаях:

- стороны хотят корректно завершить соединение (см. 8.1);
- в процессе выполнения протокола произошла ошибка (см. 8.2).

8.1 Оповещения закрытия соединения

Клиент и сервер должны всегда обмениваться друг с другом информацией о завершении соединения. В случае если в процессе работы протокола TLS не возникло ни одного оповещения с уровнем fatal, при окончании взаимодействия каждая сторона обязана послать сообщение закрытия соединения close_notify, которое предназначено для осуществления корректного завершения соединения.

Инициировать обмен оповещениями закрытия соединения может как клиент, так и сервер. Оповещение закрытия соединения, посланное первой стороной, сигнализирует о том, что его отправитель закрыл соединение для отправки данных и больше не пошлет ни одного сообщения в рамках данного соединения. Вторая сторона при получении оповещения закрытия соединения должна ответить первой стороне своим собственным оповещением закрытия соединения и немедленно закрыть соединение.

Первой стороне рекомендуется не дожидаться ответного оповещения закрытия соединения, прежде чем закрыть соединение для чтения данных. Любые данные, полученные после оповещения с типом close_notify, должны игнорироваться.

Рекомендуется присваивать оповещению закрытия соединения close_notify уровень warning.

8.2 Оповещения об ошибках

В случае возникновения ошибки в процессе работы протокола TLS сторона взаимодействия посылает оповещение об ошибке.

Настоящие рекомендации определяют оповещения об ошибке в соответствии с таблицами 8.2.1—8.2.3.

Таблица 8.2.1 — Оповещения об ошибке, которые могут возникнуть в процессе работы протокола Handshake

Название оповещения	Условия и порядок использования оповещения
handshake_failure	Отправитель не смог согласовать необходимые параметры безопасности. Данное оповещение всегда должно иметь уровень fatal
illegal_parameter	Поле сообщения протокола Handshake некорректно либо не согласуется с другими полями. Данное оповещение всегда должно иметь уровень fatal
unknown_ca	Сертификат УЦ не был найден или не совпал ни с одним из известных сертификатов доверенных УЦ. Данное оповещение всегда должно иметь уровень fatal
access_denied	Был получен корректный сертификат, однако при применении политики контроля доступа отправитель решил не продолжать согласование соединения. Данное оповещение всегда должно иметь уровень fatal
decrypt_error	При выполнении криптографической операции (например, при проверке подписи, проверке сообщения завершения Finished) во время работы протокола Handshake возникла ошибка. Данное оповещение всегда должно иметь уровень fatal
protocol_version	Версия протокола, указанная клиентом, распознана корректно, однако не поддерживается сервером. Данное оповещение всегда должно иметь уровень fatal
insufficient_security	Посылается вместо оповещения handshake_failure в том случае, если сервер требует криптонаборы, обеспечивающие более высокий уровень стойкости, чем те, что поддерживаются клиентом. Данное оповещение всегда должно иметь уровень fatal
unsupported_extension	Данное оповещение посылается клиентом и сигнализирует о том, что в сообщении ServerHello содержалось расширение, которое не было послано клиентом в сообщении ClientHello. Данное оповещение всегда должно иметь уровень fatal
bad_certificate	Сертификат был поврежден, сертификат содержит некорректную подпись и т. п. Рекомендуется присваивать данному оповещению уровень fatal
unsupported_certificate	Был получен сертификат неподдерживаемого типа. Рекомендуется присваивать данному оповещению уровень fatal
certificate_revoked	Сертификат был отозван подписывающей стороной. Рекомендуется присваивать данному оповещению уровень fatal
certificate_expired	Срок действия сертификата истек или сертификат не действует в настоящее время. Рекомендуется присваивать данному оповещению уровень fatal
certificate_unknown	При обработке сертификата возникла некоторая ошибка, не соответствующая никаким из случаев, перечисленным выше. Рекомендуется присваивать данному оповещению уровень fatal
user_canceled	Работа протокола Handshake была остановлена по некоторым причинам, не связанным с ошибками в работе протокола. Данное оповещение должно сопровождаться оповещением close_notify. Рекомендуется присваивать данному оповещению уровень warning

Окончание таблицы 8.2.1

Название оповещения	Условия и порядок использования оповещения
no_renegotiation	Данное оповещение посылается клиентом в ответ на сообщение HelloRequest или сервером в ответ на сообщение ClientHello при взаимодействии в рамках ранее установленного соединения с целью оповещения получателя о том, что отправитель данного оповещения не поддерживает пересогласование параметров сессии. При получении данного оповещения вторая сторона должна принять решение о продолжении взаимодействия. Данное оповещение всегда должно иметь уровень warning

В случае если оповещения bad_certificate, unsupported_certificate, certificate_revoked, certificate_expired, certificate_unknown были посланы с уровнем warning, непосредственно после их получения принимающая сторона должна закрыть соединение с оповещением close_notify либо закрыть соединение с оповещением об ошибке, имеющим уровень fatal.

Таблица 8.2.2 — Оповещения об ошибке, которые могут возникнуть в процессе работы протокола Record

Название оповещения	Условия и порядок использования оповещения
bad_record_mac	Данное оповещение посылается в том случае, если при обработке полученной записи возникла ошибка при проверке значения кода аутентификации. Данное оповещение всегда должно иметь уровень fatal
record_overflow	Данное оповещение посылается в том случае, если длина полученной записи превышает максимально допустимое значение. Данное оповещение всегда должно иметь уровень fatal

Таблица 8.2.3 — Общие оповещения об ошибке, которые могут возникать в процессе работы протокола TLS 1.2 в соответствии с криптонаборами, описанными в данных рекомендациях

Название оповещения	Условия и порядок использования оповещения
unexpected_message	Было получено некорректное сообщение. Данное оповещение всегда должно иметь уровень fatal
decode_error	Сообщение не может быть обработано, так как содержит одно или несколько некорректных полей. Данное оповещение всегда должно иметь уровень fatal
internal_error	Произошла внутренняя ошибка, не связанная с работой протокола (например, ошибка выделения памяти), которая делает невозможным продолжение дальнейшей работы протокола. Данное оповещение всегда должно иметь уровень fatal

9 Протокол Application Data

Прикладные данные передаются после отправки стороной взаимодействия сообщения Finished. Прикладные данные защищаются в соответствии с текущим состоянием соединения, которое было установлено в результате выполнения протокола Handshake. Сообщения протокола Application Data содержат фрагмент данных, принятых от протокола верхнего уровня, и передаются протоколу Record без дополнительного форматирования.

10 Криптонаборы

Настоящие рекомендации определяют два криптонабора, предназначенных для использования в протоколе TLS 1.2, со следующими приватными номерами:

- TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC, {0xFF, 0x89,};
- TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC, {0xFF, 0x88}.

Указанный порядок следования криптонаборов является рекомендуемым порядком предпочтения для клиентов, поддерживающих работу с обоими криптонаборами.

10.1 Сертификаты сторон

Клиент и сервер, взаимодействующие в соответствии с описанными в настоящем документе криптонаборами, могут поддерживать любой тип сертификатов для любого криптонабора.

10.2 Блочный шифр

Определенные в данных рекомендациях криптонаборы в качестве блочного шифра используют шифр «Магма» или «Кузнечик», определенные в ГОСТ Р 34.12—2015. Длина блока n составляет 16 байт (128 бит) для шифра «Кузнечик» и 8 байт (64 бита) для шифра «Магма», длина ключей k в обоих случаях составляет 32 байта (256 бит).

Таблица 10.2.1 — Алгоритм блочного шифрования

Криптонабор	Блочный шифр
TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC	ГОСТ Р 34.12-2015, «Магма»
TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC	ГОСТ Р 34.12-2015, «Кузнечик»

10.3 Алгоритм выработки кода аутентификации сообщения

Определенные в данных рекомендациях криптонаборы в качестве функции *MAC* используют блочный шифр, задающийся в соответствии с 10.2, в режиме выработки имитовставки, определенном в ГОСТ Р 34.13—2015, с длиной имитовставки равной n.

10.4 Алгоритм шифрования сообщений

Определенные в данных рекомендациях криптонаборы в качестве функции *ENC* используют блочный шифр, задающийся в соответствии с 10.2, в режиме CTR-ACPKM, определенном в Р 1323565.1.017—2018. Длина блока гаммы s, определенная в Р 1323565.1.017—2018, равна n.

Параметры режима шифрования CTR-ACPKM определены в таблице 10.4.1.

Таблица 10.4.1 — Алгоритм шифрования сообщений

Криптонабор	Размер секции
TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC	1 Кбайт
TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC	4 Кбайт

10.5 Максимальное количество записей в соединении

Параметр *SNMAX* задает максимальное количество записей, которые могут передаваться в рамках одного соединения (номер записи *seqnum* может принимать значения от 0 до *SNMAX* включительно), и определяется в соответствии с таблицей 10.5.1.

Таблица 10.5.1 — Максимальное количество записей в соединении

Криптонабор	*SNMAX*
TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC	$2^{32}-1$
TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC	$2^{64}-1$

10.6 Параметры ключевого дерева

Константы C_1, C_2, C_3, используемые для порождения ключей защиты записей, определяются в соответствии с таблицей 10.6.1.

Таблица 10.6.1 — Параметры ключевого дерева

Криптонабор	Константы C_1, C_2, C_3
TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC	$C_1 = 0xFFFFFFC000000000$, $C_2 = 0xFFFFFFFFFE000000$, $C_3 = 0xFFFFFFFFFFFFF000$
TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC	$C_1 = 0xFFFFFFFF00000000$, $C_2 = 0xFFFFFFFFFFF80000$, $C_3 = 0xFFFFFFFFFFFFFFC0$

11 Вопросы безопасности

В целях создания эффективной реализации при использовании алгоритма TLSTREE обращение к функции $Divers_j$, $j \in \{1,2,3\}$, должно проводиться только в тех случаях, когда номер записи *seqnum* достигает такого значения, что $seqnum \& C_j \neq (seqnum - 1) \& C_j$, в противном случае необходимо использовать значение, выработанное ранее. Данный подход позволяет также противодействовать атакам по побочным каналам.

Приложение А
(справочное)

Контрольные примеры работы алгоритма TLSTREE

Данное приложение носит справочный характер и не является частью настоящих рекомендаций.

А.1 TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC

Значения корневого ключа K_{root}:

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF;
```

Выработка ключа защиты записи с номером 0:

1. Ключ первого уровня, являющийся результатом работы функции $Divers_1$:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции $Divers_2$:

```
AB 3D FD 55 68 6D 55 97 63 5C FE 08 6C 12 F0 2D
40 72 2B A4 65 9F 51 77 34 F8 C5 84 B8 F9 E9 9B.
```

3. Итоговое значение ключа, являющееся результатом работы функции $Divers_3$:

```
50 76 42 D9 58 C5 20 C6 D7 EE F5 CA 8A 53 16 D4
F3 4B 85 5D 2D D4 BC BF 4E 5B F0 FF 64 1A 19 FF.
```

Выработка ключа защиты записи с номером 4095:

1. Ключ первого уровня, являющийся результатом работы функции $Divers_1$:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции $Divers_2$:

```
AB 3D FD 55 68 6D 55 97 63 5C FE 08 6C 12 F0 2D
40 72 2B A4 65 9F 51 77 34 F8 C5 84 B8 F9 E9 9B.
```

3. Итоговое значение ключа, являющееся результатом работы функции $Divers_3$:

```
50 76 42 D9 58 C5 20 C6 D7 EE F5 CA 8A 53 16 D4
F3 4B 85 5D 2D D4 BC BF 4E 5B F0 FF 64 1A 19 FF.
```

Выработка ключа защиты записи с номером 4096:

1. Ключ первого уровня, являющийся результатом работы функции $Divers_1$:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции $Divers_2$:

```
AB 3D FD 55 68 6D 55 97 63 5C FE 08 6C 12 F0 2D
40 72 2B A4 65 9F 51 77 34 F8 C5 84 B8 F9 E9 9B.
```

3. Итоговое значение ключа, являющееся результатом работы функции $Divers_3$:

```
71 86 93 EC D3 F1 DC E2 11 93 BD 56 40 6A A4 E2
09 0F E1 C8 7A 5D 5D 7A 8C E6 E0 D8 28 A4 39 15.
```

Выработка ключа защиты записи с номером 33554431:

1. Ключ первого уровня, являющийся результатом работы функции $Divers_1$:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции $Divers_2$:

```
AB 3D FD 55 68 6D 55 97 63 5C FE 08 6C 12 F0 2D
40 72 2B A4 65 9F 51 77 34 F8 C5 84 B8 F9 E9 9B.
```

3. Итоговое значение ключа, являющееся результатом работы функции $Divers_3$:

```
CD 4E 6F 09 94 C4 CA 17 A9 AD E5 3D 69 20 49 77
57 87 3B C0 8D 4F 98 06 E3 C9 99 A7 E4 AE 70 E0.
```

Выработка ключа защиты записи с номером 33554432:
1. Ключ первого уровня, являющийся результатом работы функции $Divers_1$:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции $Divers_2$:

```
AC 22 FD 64 6B 15 94 8B 94 52 EA 19 3F 40 72 C7
72 CA 54 DA 8E A2 F6 96 92 98 1F 7C 9E 29 B8 B9.
```

3. Итоговое значение ключа, являющееся результатом работы функции $Divers_3$:

```
4E AB 0F 8A 28 2D E5 78 85 7B E0 15 D1 32 CD FD
88 F0 EB D9 9D 10 CC A3 B6 D9 11 4E 37 56 5D E7.
```

Выработка ключа защиты записи с номером 274877906943:
1. Ключ первого уровня, являющийся результатом работы функции $Divers_1$:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции $Divers_2$:

```
80 A8 35 D6 A3 8A CB 77 CB AC DF C8 E8 E1 50 97
C5 8F 24 08 D6 08 B2 1B 39 CC 85 E4 1C 5C E9 5A.
```

3. Итоговое значение ключа, являющееся результатом работы функции $Divers_3$:

```
37 06 B1 BE E6 F3 BE DB 74 66 4E 1C 43 40 B5 E1
81 5F FA 87 EA 80 F4 0E AF CC 28 EF 30 DD F3 BB.
```

Выработка ключа защиты записи с номером 274877906944:
1. Ключ первого уровня, являющийся результатом работы функции $Divers_1$:

```
89 AB 52 9B 2F E3 DC 74 C2 78 46 9F 15 7F AF 6B
A9 FB 32 D4 B3 C8 A1 0E C9 DE 68 E5 2A 25 3D DE.
```

2. Ключ второго уровня, являющийся результатом работы функции $Divers_2$:

```
AA 4D 07 83 68 C7 D8 85 3C 77 BC EF C2 B5 77 FE
EC EF 53 79 FB 6A 69 22 71 5F 6D 70 5F B9 9F 3D.
```

3. Итоговое значение ключа, являющееся результатом работы функции $Divers_3$:

```
9B E5 7C B5 2F 46 B6 0C 68 40 CD 4E 95 9A 16 8A
74 82 2C 1D 00 6A C7 46 DA BB B9 84 8E B0 A5 8A.
```

A.2 TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC

Значения корневого ключа K_{root}:

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF;
```

Выработка ключа защиты записи с номером 0:
1. Ключ первого уровня, являющийся результатом работы функции $Divers_1$:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции $Divers_2$:

```
AB 3D FD 55 68 6D 55 97 63 5C FE 08 6C 12 F0 2D
40 72 2B A4 65 9F 51 77 34 F8 C5 84 B8 F9 E9 9B.
```

3. Итоговое значение ключа, являющееся результатом работы функции $Divers_3$:

```
50 76 42 D9 58 C5 20 C6 D7 EE F5 CA 8A 53 16 D4
F3 4B 85 5D 2D D4 BC BF 4E 5B F0 FF 64 1A 19 FF.
```

Выработка ключа защиты записи с номером 63:
1. Ключ первого уровня, являющийся результатом работы функции ***Divers*$_1$**:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции ***Divers*$_2$**:

```
AB 3D FD 55 68 6D 55 97 63 5C FE 08 6C 12 F0 2D
40 72 2B A4 65 9F 51 77 34 F8 C5 84 B8 F9 E9 9B.
```

3. Итоговое значение ключа, являющееся результатом работы функции ***Divers*$_3$**:

```
50 76 42 D9 58 C5 20 C6 D7 EE F5 CA 8A 53 16 D4
F3 4B 85 5D 2D D4 BC BF 4E 5B F0 FF 64 1A 19 FF.
```

Выработка ключа защиты записи с номером 64:
1. Ключ первого уровня, являющийся результатом работы функции ***Divers*$_1$**:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции ***Divers*$_2$**:

```
AB 3D FD 55 68 6D 55 97 63 5C FE 08 6C 12 F0 2D
40 72 2B A4 65 9F 51 77 34 F8 C5 84 B8 F9 E9 9B.
```

3. Итоговое значение ключа, являющееся результатом работы функции ***Divers*$_3$**:

```
C1 9B 63 9F 4B EA 78 8C 1C 59 B4 C8 87 DB 5B 07
C1 91 19 10 18 68 DA B8 9A 8D 93 61 B2 F0 10 F3.
```

Выработка ключа защиты записи с номером 524287:
1. Ключ первого уровня, являющийся результатом работы функции ***Divers*$_1$**:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции ***Divers*$_2$**:

```
AB 3D FD 55 68 6D 55 97 63 5C FE 08 6C 12 F0 2D
40 72 2B A4 65 9F 51 77 34 F8 C5 84 B8 F9 E9 9B.
```

3. Итоговое значение ключа, являющееся результатом работы функции ***Divers*$_3$**:

```
92 30 3E B5 61 56 88 54 E3 3E 4F E0 97 A9 95 99
17 9F 5B 90 94 AE 34 79 E6 1C 43 69 3A 3F 0A 06.
```

Выработка ключа защиты записи с номером 524288:
1. Ключ первого уровня, являющийся результатом работы функции ***Divers*$_1$**:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции ***Divers*$_2$**:

```
1E C6 92 34 D7 1F B8 92 1A CF 26 C9 2F B8 8C 52
10 57 0B 6B 21 0D BF 77 4E 75 97 C1 D8 CE 16 0C.
```

3. Итоговое значение ключа, являющееся результатом работы функции ***Divers*$_3$**:

```
E8 55 A0 E2 CB DD 68 C1 13 7C EF 3E 80 1E 0B FF
68 62 8C 36 43 68 27 6D 0C B8 7E B5 6E 94 EF 42.
```

Выработка ключа защиты записи с номером 4294967295:
1. Ключ первого уровня, являющийся результатом работы функции ***Divers*$_1$**:

```
27 67 9F FB FA 99 43 54 30 51 FD 5E A3 50 C7 F0
32 EC C2 E7 5F A2 11 96 A6 AE 12 C0 14 83 F1 A2.
```

2. Ключ второго уровня, являющийся результатом работы функции ***Divers*$_2$**:

```
ED 6A F8 70 1B 59 A1 EF D4 5A 6E 33 CF 80 B7 66
D5 20 7A 0B 28 80 CD 63 63 4E 80 B1 83 F9 F3 D4.
```

3. Итоговое значение ключа, являющееся результатом работы функции ***Divers*$_3$**:

```
93 D3 CA E9 5A 55 B7 1A A3 B9 A7 DD F9 9A 6A AC
3F DA 17 2A 79 60 58 04 A9 C9 FC 6E 84 8A F1 AA.
```

Выработка ключа защиты записи с номером 4294967296:

1. Ключ первого уровня, являющийся результатом работы функции ***Divers*$_1$**:

```
E2 BE DA BC 66 55 9F 65 FB 55 E5 52 73 56 50 EE
71 0F 08 81 A0 81 EF EA 6B 7A BA 82 BD AF 8A 6E.
```

2. Ключ второго уровня, являющийся результатом работы функции ***Divers*$_2$**:

```
5E 6A D9 06 EE E7 85 F8 BE 28 6C C7 04 C2 5D 46
D6 54 9D 18 71 9D 9B E7 FE 79 D0 1D A4 37 A1 DB.
```

3. Итоговое значение ключа, являющееся результатом работы функции ***Divers*$_3$**:

```
7F FB 1A D7 E5 7B 70 BE 10 96 31 D2 71 92 98 B9
7D EE 3B 00 8D 86 F8 3D AA F6 2A 4E A5 B7 AA FD.
```

Приложение Б
(справочное)

Контрольные примеры для работы протокола Record

Данное приложение носит справочный характер и не является частью настоящих рекомендаций.

Так как процесс обработки данных во время работы протокола Record одинаков для сервера и клиента, все действия описываются для одной фиксированной стороны взаимодействия. Способ фрагментации данных, длина данных и количество записей выбраны из условий удобства и наглядности.

Для наглядности байтовые строки разделяются на подстроки (не более 16 байтов в подстроке) с указанием смещения текущей подстроки относительно начала строки. Смещение указывается слева от подстроки в шестнадцатеричном виде и отделяется от соответствующей подстроки двоеточием.

Б.1 TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC

В настоящем разделе приведен тестовый пример работы протокола Record для криптонабора TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC. Рассматривается случай обработки 4097 сообщений, при этом значения контрольных примеров приводятся для трех из них, демонстрирующих наиболее важные особенности работы протокола.

Предполагается, что в результате выполнения протокола Handshake стороной взаимодействия были выработаны следующие параметры:

- ключ вычисления имитовставки сообщения K_{MAC}:

```
000000:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000010:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF;
```

- ключ зашифрования данных K_{ENC}:

```
000000:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00;
```

- вектор инициализации для зашифрования данных IV:

```
000000:  00 00 00 00.
```

Б.1.1 Формирование защищенной записи с номером 0

Фрагмент данных длины 7 байт, передающийся в рамках протокола Application Data в записи с номером $seqnum = 0$:

```
000000:  00 00 00 00 00 00 00.
```

Незащищенная запись Rec^0, представленная в виде структуры TLSPlaintext:

```
000000:  17 03 03 00 07 00 00 00 00 00 00 00,
```

Ключ K^0_{MAC} вычисления имитовставки для записи с номером $seqnum = 0$, выработанный в результате работы алгоритма TLSTREE:

```
000000:  50 76 42 D9 58 C5 20 C6 D7 EE F5 CA 8A 53 16 D4
000010:  F3 4B 85 5D 2D D4 BC BF 4E 5B F0 FF 64 1A 19 FF.
```

Значение $RecMAC$ для записи с номером $seqnum = 0$:

```
000000:  30 70 B8 64 77 9D 95 47.
```

Ключ K^0_{ENC} зашифрования для записи с номером $seqnum = 0$, выработанный в результате работы алгоритма TLSTREE:

```
000000:  F7 97 25 68 45 F3 6C F0 75 60 34 45 CD 32 2B AC
000010:  C3 83 40 32 BC 42 5E 4D 3C 84 95 23 6F 7B 6C AF.
```

Вектор инициализации IV^0 для записи с номером $seqnum = 0$:

```
000000:  00 00 00 00.
```

Зашифрование данных $ENCData$ проводится на одном секционном ключе K^1.

Секционный ключ K^1:

```
000000:  F7 97 25 68 45 F3 6C F0 75 60 34 45 CD 32 2B AC
000010:  C3 83 40 32 BC 42 5E 4D 3C 84 95 23 6F 7B 6C AF.
```

Защищенная запись $PRec^0$, представленная в виде структуры TLSCiphertext:

```
000000:  17 03 03 00 0F 1B B1 C2 20 BE 6D 55 F4 79 58 DD
000010:  63 81 F3 D6.
```

Б.1.2 Формирование защищенной записи с номером 4095

Фрагмент данных длины 1024 байта (1 Кбайт), передающийся в рамках протокола Application Data в записи с номером $seqnum = 4095$:

```
000000:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
0003D0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0003E0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0003F0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.
```

Незащищенная запись Rec^{4095}, представленная в виде структуры TLSPlaintext:

```
000000:  17 03 03 04 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
0003D0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0003E0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0003F0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000400:  00 00 00 00 00.
```

Ключ K_{MAC}^{4095} вычисления имитовставки для записи с номером $seqnum = 4095$ равен ключу K_{MAC}^{0}:

```
000000:  50 76 42 D9 58 C5 20 C6 D7 EE F5 CA 8A 53 16 D4
000010:  F3 4B 85 5D 2D D4 BC BF 4E 5B F0 FF 64 1A 19 FF.
```

Значение $RecMAC$ для записи с номером $seqnum = 4095$:

```
000000:  ED EE A5 8C E6 28 B7 7C.
```

Ключ K_{ENC}^{4095} зашифрования для записи с номером $seqnum = 4095$ равен ключу K_{ENC}^{0}:

```
000000:  F7 97 25 68 45 F3 6C F0 75 60 34 45 CD 32 2B AC
000010:  C3 83 40 32 BC 42 5E 4D 3C 84 95 23 6F 7B 6C AF.
```

Вектор инициализации IV^{4095} для записи с номером $seqnum = 4095$:

```
000000:  00 00 0F FF.
```

Данные $ENCData$ разбиваются на две секции, зашифрование проводится на двух секционных ключах K^1 и K^2. Секционный ключ K^1:

```
000000:  F7 97 25 68 45 F3 6C F0 75 60 34 45 CD 32 2B AC
000010:  C3 83 40 32 BC 42 5E 4D 3C 84 95 23 6F 7B 6C AF.
```

Секционный ключ K^2:

```
000000:  44 90 5F EF 56 6A 28 2A B0 71 A6 C4 FB 54 84 06
000010:  10 43 10 22 A0 5C 76 47 32 1A 15 4E 3E 2F 03 67.
```

Защищенная запись $PRec^{4095}$, представленная в виде структуры TLSCiphertext:

```
000000:  17 03 03 04 08 A2 DE 3A B5 2F DA C7 5C FE 84 E4
000010:  39 92 35 48 51 73 91 7D FE 7F 59 03 0A A0 B0 55
000020:  09 16 64 74 E2 50 01 7C 8F B6 89 77 BB BE 38 BB
. . .
0003D0:  67 25 5B 60 D5 0B CA 42 12 51 92 C1 A3 F8 94 BC
0003E0:  ED 93 3B 50 3D 7A 1F 2F D7 72 B1 76 FD 52 B8 EC
0003F0:  A3 E6 91 F1 D8 CC 9C 52 35 75 25 E1 99 30 AB 48
000400:  4D 9B 42 42 95 E3 18 64 25 D7 FD B6 90.
```

Б.1.3 Формирование защищенной записи с номером 4096

Фрагмент данных длины 2048 байт (2 Кбайта), передающийся в рамках протокола Application Data в записи с номером $seqnum = 4096$:

```
000000:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
0007D0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007E0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007F0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.
```

Незащищенная запись Rec^{4096}, представленная в виде структуры TLSPlaintext:

```
000000:  17 03 03 08 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
0007D0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007E0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007F0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000800:  00 00 00 00 00.
```

Ключ K_{MAC}^{4096} вычисления имитовставки для записи с номером $seqnum = 4096$, выработанный в результате работы алгоритма TLSTREE:

```
000000:  71 86 93 EC D3 F1 DC E2 11 93 BD 56 40 6A A4 E2
000010:  09 0F E1 C8 7A 5D 5D 7A 8C E6 E0 D8 28 A4 39 15.
```

Значение $RecMAC$ для записи с номером $seqnum = 4096$:

```
000000:  9E 76 D1 8E 75 A5 DB 32.
```

Ключ K_{ENC}^{4096} зашифрования для записи с номером $seqnum = 4096$, выработанный в результате работы алгоритма TLSTREE:

```
000000:  60 BA 8C DF 9C 67 0A D0 5C 5B E3 B5 E0 04 B7 37
000010:  13 9F 1A A9 05 92 8A 02 8F 6C C4 B0 95 49 CC A4.
```

Вектор инициализации IV^{4096} для записи с номером $seqnum = 4096$:

```
000000:  00 00 10 00.
```

Данные $ENCData$ разбиваются на три секции, зашифрование проводится на трех секционных ключах K^1, K^2 и K^3.

Секционный ключ K^1:

```
000000:  60 BA 8C DF 9C 67 0A D0 5C 5B E3 B5 E0 04 B7 37
000010:  13 9F 1A A9 05 92 8A 02 8F 6C C4 B0 95 49 CC A4.
```

Секционный ключ K^2:

```
000000:  08 B0 0A BB 92 9E D2 EE 8F 73 3B 95 DF 1B D4 F5
000010:  84 24 A1 8A 37 53 0C BB 34 E7 F7 AF 50 09 9B 51.
```

Секционный ключ K^3:

```
000000:  74 E2 E2 A9 C3 3A 22 D2 20 43 F8 33 0A 46 DC A7
000010:  54 0D D0 6E EC 1E 3B 0B 06 B8 D3 E2 EE 4A 22 95.
```

Защищенная запись $PRec^{4096}$, представленная в виде структуры TLSCiphertext:

```
000000:  17 03 03 08 08 37 97 76 53 95 C2 35 18 98 17 23
000010:  CA B5 85 4F 7A 90 9C F5 D0 B7 9F 5B 6E D1 6C 65
000020:  1B AA 29 C6 CD DC 32 06 5F 75 BD 67 C8 CE 08 8A
. . .
0007D0:  0A F0 56 4B B9 E1 1E 82 D5 A2 42 D3 24 F2 5F 72
0007E0:  8D C8 07 1F 4A AD 14 17 F7 9A 8D BB 8B AE 57 CF
0007F0:  D2 2C E6 72 1C 26 E6 4E 66 2A 3B 5F D3 80 4D 27
000800:  1A B0 41 A6 2E 89 DB BB 8F 24 92 AE 5C.
```

Б.2 TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC

В настоящем разделе приведен тестовый пример работы протокола Record для криптонабора TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC. Рассматривается случай обработки 65 сообщений,

при этом значения контрольных примеров приводятся для трех из них, демонстрирующих наиболее важные особенности работы протокола.

Предполагается, что в результате выполнения протокола Handshake стороной взаимодействия были выработаны следующие параметры:

- ключ вычисления имитовставки сообщения K_{MAC}:

```
000000:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000010:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF;
```

- ключ зашифрования данных K_{ENC}:

```
000000:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00;
```

- вектор инициализации для зашифрования данных *IV*:

```
000000:  00 00 00 00 00 00 00 00.
```

Б.2.1 Формирование защищенной записи с номером 0

Фрагмент данных длины 15 байт, передающийся в рамках протокола Application Data в записи с номером *seqnum* = 0:

```
000000:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.
```

Незащищенная запись Rec^0, представленная в виде структуры TLSPlaintext:

```
000000:  17 03 03 00 0F 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00.
```

Ключ K^0_{MAC} вычисления имитовставки для записи с номером *seqnum* = 0, выработанный в результате работы алгоритма TLSTREE:

```
000000:  50 76 42 D9 58 C5 20 C6 D7 EE F5 CA 8A 53 16 D4
000010:  F3 4B 85 5D 2D D4 BC BF 4E 5B F0 FF 64 1A 19 FF.
```

Значение *RecMAC* для записи с номером *seqnum* = 0:

```
000000:  75 53 09 CB C7 3B B9 49 C5 0E BB 86 16 0A 0F EE.
```

Ключ K^0_{ENC} зашифрования для записи с номером *seqnum* = 0, выработанный в результате работы алгоритма TLSTREE:

```
000000:  F7 97 25 68 45 F3 6C F0 75 60 34 45 CD 32 2B AC
000010:  C3 83 40 32 BC 42 5E 4D 3C 84 95 23 6F 7B 6C AF.
```

Вектор инициализации IV^0 для записи с номером *seqnum* = 0:

```
000000:  00 00 00 00 00 00 00 00.
```

Зашифрование данных *ENCData* проводится на одном секционном ключе K^1.

Секционный ключ K^1:

```
000000:  F7 97 25 68 45 F3 6C F0 75 60 34 45 CD 32 2B AC
000010:  C3 83 40 32 BC 42 5E 4D 3C 84 95 23 6F 7B 6C AF.
```

Защищенная запись $PRec^0$, представленная в виде структуры TLSCiphertext:

```
000000:  17 03 03 00 1F F3 17 A7 1D 3A CE 43 3B 01 D4 E7
000010:  D4 EF 61 AE 00 D5 3B 41 52 7A 26 1E DF C2 BA 78
000020:  57 C1 93 2D.
```

Б.2.2 Формирование защищенной записи с номером 63

Фрагмент данных длины 4096 байт (4 Кбайта), передающийся в рамках протокола Application Data в записи с номером *seqnum* = 63:

```
000000:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
000FD0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000FE0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000FF0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.
```

Незащищенная запись Rec^{63}, представленная в виде структуры TLSPlaintext:

```
000000:  17 03 03 10 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
000FD0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000FE0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000FF0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001000:  00 00 00 00 00.
```

Ключ K_{MAC}^{63} вычисления имитовставки для записи с номером $seqnum = 63$ равен ключу K_{MAC}^{0}:

```
000000:  50 76 42 D9 58 C5 20 C6 D7 EE F5 CA 8A 53 16 D4
000010:  F3 4B 85 5D 2D D4 BC BF 4E 5B F0 FF 64 1A 19 FF.
```

Значение *RecMAC* для записи с номером $seqnum = 63$:

```
000000:  0A 3B FD 43 0F CD D8 D8 5C 96 46 86 81 78 4F 7D.
```

Ключ K_{ENC}^{63} зашифрования для записи с номером $seqnum = 63$ равен ключу K_{ENC}^{0}:

```
000000:  F7 97 25 68 45 F3 6C F0 75 60 34 45 CD 32 2B AC
000010:  C3 83 40 32 BC 42 5E 4D 3C 84 95 23 6F 7B 6C AF.
```

Вектор инициализации IV^{63} для записи с номером $seqnum = 63$:

```
000000:  00 00 00 00 00 00 00 3F.
```

Данные *ENCData* разбиваются на две секции, зашифрование проводится на двух секционных ключах K^1 и K^2. Секционный ключ K^1:

```
000000:  F7 97 25 68 45 F3 6C F0 75 60 34 45 CD 32 2B AC
000010:  C3 83 40 32 BC 42 5E 4D 3C 84 95 23 6F 7B 6C AF.
```

Секционный ключ K^2:

```
000000:  3E E3 23 10 86 B9 3E 86 69 3B 6A 08 25 ED 68 4B
000010:  2B CC D2 89 E4 E7 6C 7D 4D B8 89 AA E2 DA DC DD.
```

Защищенная запись $PRec^{63}$, представленная в виде структуры TLSCiphertext:

```
000000:  17 03 03 10 10 6A 18 38 B0 A0 D5 A0 4D 1F 29 64
000010:  89 6D 08 5F B7 DA 84 D7 76 C3 9F 5C DC 37 20 B7
000020:  B5 59 EF 13 9D E4 38 F8 84 97 BD 27 92 A1 67 31
. . .
000FE0:  82 5C B1 24 41 0A 81 29 9B 35 98 19 5D D4 51 68
000FF0:  A6 38 50 A7 6E 1A 4F 1E 6D D5 EF 72 59 3F AE 76
001000:  55 71 EC 37 E7 17 F5 B8 62 85 BB 5B FD 83 B6 6A
001010:  B7 63 86 52 08.
```

Б.2.3 Формирование защищенной записи с номером 64

Фрагмент данных длины 8192 байта (8 Кбайт), передающийся в рамках протокола Application Data в записи с номером $seqnum = 64$:

```
000000:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
001FD0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001FE0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001FF0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.
```

Незащищенная запись Rec^{64}, представленная в виде структуры TLSPlaintext:

```
000000:  17 03 03 20 00 00 00 00 00 00 00 00 00 00 00 00
000010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
```

```
001FD0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001FE0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001FF0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002000:  00 00 00 00 00.
```

Ключ K_{MAC}^{64} вычисления имитовставки для записи с номером $seqnum = 64$, выработанный в результате работы алгоритма TLSTREE:

```
000000:  C1 9B 63 9F 4B EA 78 8C 1C 59 B4 C8 87 DB 5B 07
000010:  C1 91 19 10 18 68 DA B8 9A 8D 93 61 B2 F0 10 F3.
```

Значение RecMAC для записи с номером $seqnum = 64$:

```
000000:  47 25 A2 3C 64 3C CA E1 28 28 FE DC 0D 65 EB B3.
```

Ключ K_{ENC}^{64} зашифрования для записи с номером $seqnum = 64$, выработанный в результате работы алгоритма TLSTREE:

```
000010:  F3 C9 BA 5C 5D 43 AA 38 8B 9F 9A 8C 98 05 23 9D
000010:  16 23 08 C8 85 CB 2D 1C 9A 76 13 83 2F 96 70 00.
```

Вектор инициализации IV^{64} для записи с номером $seqnum = 64$:

```
000000:  00 00 00 00 00 00 00 40.
```

Данные *ENCData* разбиваются на три секции, зашифрование проводится на трех секционных ключах K^1, K^2 и K^3.

Секционный ключ K^1:

```
000000:  F3 C9 BA 5C 5D 43 AA 38 8B 9F 9A 8C 98 05 23 9D
000010:  16 23 08 C8 85 CB 2D 1C 9A 76 13 83 2F 96 70 00.
```

Секционный ключ K^2:

```
000000:  F8 40 F3 80 A0 7B B6 2D EC 02 5F 9A 07 46 AF 68
000010:  EC 19 23 75 3F F6 16 24 D5 84 A5 53 DD 76 FB 78.
```

Секционный ключ K^3:

```
000000:  A9 3D 06 1E 9B 9E 0C 88 02 2E E4 A3 8B FD 4B 37
000010:  81 20 18 B2 8A 3E 99 DE 96 FD 71 4D 55 18 A5 BC.
```

Защищенная запись $PRec^{64}$, представленная в виде структуры TLSCiphertext:

```
000000:  17 03 03 20 10 FD 8D 03 90 D0 28 1A 18 CF 0D 24
000010:  9A 83 A8 05 71 A2 CD DA 64 E4 E6 9E FE 11 4A B0
000020:  4F 24 5C 6A 30 E9 F1 80 28 9A D6 09 7E F8 C1 81
. . .
001FE0:  1F 03 21 53 6E 1E FC E3 5E 75 EC 9C AE 8A 0D 17
001FF0:  C6 73 B1 72 15 BC 2D CE CB 35 E9 52 6D 14 6D 6E
002000:  25 4E A9 DA EA A4 B0 4A 73 AB BA 93 12 FE D0 B1
002010:  93 66 89 ED AB.
```

Приложение В
(справочное)

Контрольные примеры для работы протокола TLS

Данное приложение носит справочный характер и не является частью настоящих рекомендаций.

Для наглядности данные, которыми оперируют клиент и сервер, представлены слева и справа от вертикальной черты соответственно. При этом обозначения «------>» и «<------» используются для указания на данные, пересылаемые по каналу связи клиентом и сервером соответственно.

B.1 TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC

В настоящем разделе приведен тестовый пример сценария работы протокола TLS 1.2 для криптонабора TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC, при котором используется полная схема работы протокола Handshake с односторонней аутентификацией. Открытый и закрытый ключи сервера задаются следующим образом.

Идентификатор кривой сертификата сервера:
id-tc26-gost-3410-2012-256-paramSetA, «1.2.643.7.1.2.1.1.1».

Открытый ключ сервера Q_S:

```
x = 0xC03ED6E3604CCB026E56E91272454707
      94E0508A1FE1602371F443DC20E313FD;
y = 0x994D579491A2DBB720E78E0AFAC8479C
      8D1A852989A46D4969C799E39A1025EC.
```

Закрытый ключ сервера k_S:

```
0x0880FD1E94C7656CA143A310553A04B5
  5583E16ABF56D91336AB271B7396E0E4.
```

Предполагается, что соединение устанавливается впервые. Для удобства в рамках каждой записи передается только одно сообщение.

Формирование сообщения ClientHello на стороне клиента:

```
msg_type:                 01
length:                   000040
body:
  client_version:
    major:                03
    minor:                03
  random:                 933EA21EC3802A561550EC78D6ED51AC
                          2439D7E749C31BC3A3456165889684CA
  session_id:
    length:               00
    vector:               --
  cipher_suites:
    length:               0004
    vector:
      CipherSuite:        FF88
      CipherSuite:        FF89
  compression_methods:
    length:               01
    vector:
      CompressionMethod:  00
  extensions:
    length:               0013
    vector:
      Extension: /* signature_algorithms */
        extension_type:   000D
        extension_data:
          length:         0006
          vector:
            supported_signature_algorithms:
              length:     0004
              vector:
```

```
                /* Первая пара алгоритмов */
                hash:    EE
                signature:
                         EE
                /* Вторая пара алгоритмов */
                hash:    EF
                signature:
                         EF
        Extension: /* renegotiation_info */
          extension_type:  FF01
          extension_data:
            length:        0001
            vector:
              renegotiated_connection:
                length:    00
                vector:    --
        Extension: /* extended_master_secret */
          extension_type:  0017
          extension_data:
            length:        0000
            vector:        --
```

Итоговое сообщение ClientHello:

```
000000:   01 00 00 40 03 03 93 3E A2 1E C3 80 2A 56 15 50
000010:   EC 78 D6 ED 51 AC 24 39 D7 E7 49 C3 1B C3 A3 45
000020:   61 65 88 96 84 CA 00 00 04 FF 88 FF 89 01 00 00
000030:   13 00 0D 00 06 00 04 EE EE EF EF FF 01 00 01 00
000040:   00 17 00 00
```

Формирование незащищенной записи с номером 0 на стороне клиента:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 0044
fragment:               010000400303933EA21EC3802A561550
                        EC78D6ED51AC2439D7E749C31BC3A345
                        6165889684CA000004FF88FF89010000
                        13000D00060004EEEEEFEFFF01000100
                        00170000
```

Итоговая запись с номером 0 на стороне клиента:

```
000000:   16 03 03 00 44 01 00 00 40 03 03 93 3E A2 1E C3
000010:   80 2A 56 15 50 EC 78 D6 ED 51 AC 24 39 D7 E7 49
000020:   C3 1B C3 A3 45 61 65 88 96 84 CA 00 00 04 FF 88
000030:   FF 89 01 00 00 13 00 0D 00 06 00 04 EE EE EF EF
000040:   FF 01 00 01 00 00 17 00 00
```

-------->

Формирование сообщения ServerHello на стороне сервера:

```
msg_type:               02
length:                 000041
body:
  server_version:
    major:              03
    minor:              03
  random:               933EA21E49C31BC3A3456165889684CA
                        A5576CE7924A24F58113808DBD9EF856
  session_id:
    length:             10
    vector:         C3802A561550EC78D6ED51AC2439D7E7
```

```
  cipher_suite:
    CipherSuite:          FF88
  compression_method:
    CompressionMethod:    00
  extensions:
    length:               0009
    vector:
      Extension: /* renegotiation_info */
        extension_type:   FF01
        extension_data:
          length:         0001
          vector:
            renegotiated_connection:
              length:     00
              vector:     --
      Extension: /* extended_master_secret */
        extension_type:   0017
        extension_data:
          length:         0000
          vector:         --
```

Итоговое сообщение ServerHello:

```
000000:   02 00 00 41 03 03 93 3E A2 1E 49 C3 1B C3 A3 45
000010:   61 65 88 96 84 CA A5 57 6C E7 92 4A 24 F5 81 13
000020:   80 8D BD 9E F8 56 10 C3 80 2A 56 15 50 EC 78 D6
000030:   ED 51 AC 24 39 D7 E7 FF 88 00 00 09 FF 01 00 01
000040:   00 00 17 00 00
```

Формирование незащищенной записи с номером 0 на стороне сервера:

```
type:                     16
version:
  major:                  03
  minor:                  03
length:                   0045
fragment:                 020000410303933EA21E49C31BC3A345
                          6165889684CAA5576CE7924A24F58113
                          808DBD9EF85610C3802A561550EC78D6
                          ED51AC2439D7E7FF88000009FF010001
                          0000170000
```

Итоговая запись с номером 0 на стороне сервера:

```
000000:   16 03 03 00 45 02 00 00 41 03 03 93 3E A2 1E 49
000010:   C3 1B C3 A3 45 61 65 88 96 84 CA A5 57 6C E7 92
000020:   4A 24 F5 81 13 80 8D BD 9E F8 56 10 C3 80 2A 56
000030:   15 50 EC 78 D6 ED 51 AC 24 39 D7 E7 FF 88 00 00
000040:   09 FF 01 00 01 00 00 17 00 00
```

<-------

Формирование сообщения Certificate на стороне сервера:

```
msg_type:                 0B
length:                   0002BC
body:
  certificate_list:
    length:               0002B9
    vector:
      ASN.1Cert:
        length:           0002B6
        vector:           308202B230820261A003020102020A28
                          A290E3000000D8D142300806062A8503
                          020203303A3112301060A0992268993
                                                . . .
```

```
                                1D996A750AC8DA2C3EF228475ED3FBC7
                                9A3EA1C7D580AE08D081F314B48809BD
                                2CD4B58FA84CB2B66611FD6C0A84BE59
                                253D1887CC02
```

Итоговое сообщение Certificate:

```
000000:    0B 00 02 BC 00 02 B9 00 02 B6 30 82 02 B2 30 82
000010:    02 61 A0 03 02 01 02 02 0A 28 A2 90 E3 00 00 00
000020:    D8 D1 42 30 08 06 06 2A 85 03 02 02 03 30 3A 31
000030:    12 30 10 06 0A 09 92 26 89 93 F2 2C 64 01 19 16
000040:    02 72 75 31 12 30 10 06 0A 09 92 26 89 93 F2 2C
000050:    64 01 19 16 02 63 70 31 10 30 0E 06 03 55 04 03
000060:    13 07 74 65 73 74 2D 63 61 30 1E 17 0D 31 37 31
000070:    30 32 34 30 32 35 30 35 36 5A 17 0D 32 37 31 30
000080:    32 34 30 39 33 30 35 36 5A 30 21 31 1F 30 1D 06
000090:    03 55 04 03 13 16 53 65 72 76 65 72 54 4C 53 31
0000A0:    32 54 65 73 74 53 61 6D 70 6C 65 73 30 68 30 21
0000B0:    06 08 2A 85 03 07 01 01 01 01 30 15 06 09 2A 85
0000C0:    03 07 01 02 01 01 01 06 08 2A 85 03 07 01 01 02
0000D0:    02 03 43 00 04 40 FD 13 E3 20 DC 43 F4 71 23 60
0000E0:    E1 1F 8A 50 E0 94 07 47 45 72 12 E9 56 6E 02 CB
0000F0:    4C 60 E3 D6 3E C0 EC 25 10 9A E3 99 C7 69 49 6D
000100:    A4 89 29 85 1A 8D 9C 47 C8 FA 0A 8E E7 20 B7 DB
000110:    A2 91 94 57 4D 99 A3 82 01 59 30 82 01 55 30 13
000120:    06 03 55 1D 25 04 0C 30 0A 06 08 2B 06 01 05 05
000130:    07 03 01 30 0E 06 03 55 1D 0F 01 01 FF 04 04 03
000140:    02 04 F0 30 1D 06 03 55 1D 0E 04 16 04 14 B0 90
000150:    04 86 FC 71 C5 91 5A CA 9B 6B 36 1C 18 A8 37 14
000160:    35 1B 30 1F 06 03 55 1D 23 04 18 30 16 80 14 9E
000170:    03 F0 B8 9C FC 60 DC 8A 18 1E E8 00 DF A8 5B 32
000180:    CD 73 76 30 3F 06 03 55 1D 1F 04 38 30 36 30 34
000190:    A0 32 A0 30 86 2E 68 74 74 70 3A 2F 2F 76 6D 2D
0001A0:    74 65 73 74 2D 63 61 2E 63 70 2E 72 75 2F 43 65
0001B0:    72 74 45 6E 72 6F 6C 6C 2F 74 65 73 74 2D 63 61
0001C0:    2E 63 72 6C 30 81 AC 06 08 2B 06 01 05 05 07 01
0001D0:    01 04 81 9F 30 81 9C 30 4B 06 08 2B 06 01 05 05
0001E0:    07 30 02 86 3F 68 74 74 70 3A 2F 2F 76 6D 2D 74
0001F0:    65 73 74 2D 63 61 2E 63 70 2E 72 75 2F 43 65 72
000200:    74 45 6E 72 6F 6C 6C 2F 76 6D 2D 74 65 73 74 2D
000210:    63 61 2E 63 70 2E 72 75 5F 74 65 73 74 2D 63 61
000220:    2E 63 72 74 30 4D 06 08 2B 06 01 05 05 07 30 02
000230:    86 41 66 69 6C 65 3A 2F 2F 5C 5C 76 6D 2D 74 65
000240:    73 74 2D 63 61 2E 63 70 2E 72 75 5C 43 65 72 74
000250:    45 6E 72 6F 6C 6C 5C 76 6D 2D 74 65 73 74 2D 63
000260:    61 2E 63 70 2E 72 75 5F 74 65 73 74 2D 63 61 2E
000270:    63 72 74 30 08 06 06 2A 85 03 02 02 03 03 41 00
000280:    93 30 50 11 50 42 80 3B 6F DD 1D 99 6A 75 0A C8
000290:    DA 2C 3E F2 28 47 5E D3 FB C7 9A 3E A1 C7 D5 80
0002A0:    AE 08 D0 81 F3 14 B4 88 09 BD 2C D4 B5 8F A8 4C
0002B0:    B2 B6 66 11 FD 6C 0A 84 BE 59 25 3D 18 87 CC 02
```

Формирование незащищенной записи с номером 1 на стороне сервера:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 02C0
fragment:               0B0002BC0002B90002B6308202B23082
                        0261A003020102020A28A290E3000000
                        D8D142300806062A8503020203303A31
                                      . . .
```

```
                                DA2C3EF228475ED3FBC79A3EA1C7D580
                                AE08D081F314B48809BD2CD4B58FA84C
                                B2B66611FD6C0A84BE59253D1887CC02
```

Итоговая запись с номером 1 на стороне сервера:

```
000000:   16 03 03 02 C0 0B 00 02 BC 00 02 B9 00 02 B6 30
000010:   82 02 B2 30 82 02 61 A0 03 02 01 02 02 0A 28 A2
000020:   90 E3 00 00 00 D8 D1 42 30 08 06 06 2A 85 03 02
000030:   02 03 30 3A 31 12 30 10 06 0A 09 92 26 89 93 F2
000040:   2C 64 01 19 16 02 72 75 31 12 30 10 06 0A 09 92
000050:   26 89 93 F2 2C 64 01 19 16 02 63 70 31 10 30 0E
000060:   06 03 55 04 03 13 07 74 65 73 74 2D 63 61 30 1E
000070:   17 0D 31 37 31 30 32 34 30 32 35 30 35 36 5A 17
000080:   0D 32 37 31 30 32 34 30 39 33 30 35 36 5A 30 21
000090:   31 1F 30 1D 06 03 55 04 03 13 16 53 65 72 76 65
0000A0:   72 54 4C 53 31 32 54 65 73 74 53 61 6D 70 6C 65
0000B0:   73 30 68 30 21 06 08 2A 85 03 07 01 01 01 01 30
0000C0:   15 06 09 2A 85 03 07 01 02 01 01 01 06 08 2A 85
0000D0:   03 07 01 01 02 02 03 43 00 04 40 FD 13 E3 20 DC
0000E0:   43 F4 71 23 60 E1 1F 8A 50 E0 94 07 47 45 72 12
0000F0:   E9 56 6E 02 CB 4C 60 E3 D6 3E C0 EC 25 10 9A E3
000100:   99 C7 69 49 6D A4 89 29 85 1A 8D 9C 47 C8 FA 0A
000110:   8E E7 20 B7 DB A2 91 94 57 4D 99 A3 82 01 59 30
000120:   82 01 55 30 13 06 03 55 1D 25 04 0C 30 0A 06 08
000130:   2B 06 01 05 05 07 03 01 30 0E 06 03 55 1D 0F 01
000140:   01 FF 04 04 03 02 04 F0 30 1D 06 03 55 1D 0E 04
000150:   16 04 14 B0 90 04 86 FC 71 C5 91 5A CA 9B 6B 36
000160:   1C 18 A8 37 14 35 1B 30 1F 06 03 55 1D 23 04 18
000170:   30 16 80 14 9E 03 F0 B8 9C FC 60 DC 8A 18 1E E8
000180:   00 DF A8 5B 32 CD 73 76 30 3F 06 03 55 1D 1F 04
000190:   38 30 36 30 34 A0 32 A0 30 86 2E 68 74 74 70 3A
0001A0:   2F 2F 76 6D 2D 74 65 73 74 2D 63 61 2E 63 70 2E
0001B0:   72 75 2F 43 65 72 74 45 6E 72 6F 6C 6C 2F 74 65
0001C0:   73 74 2D 63 61 2E 63 72 6C 30 81 AC 06 08 2B 06
0001D0:   01 05 05 07 01 01 04 81 9F 30 81 9C 30 4B 06 08
0001E0:   2B 06 01 05 05 07 30 02 86 3F 68 74 74 70 3A 2F
0001F0:   2F 76 6D 2D 74 65 73 74 2D 63 61 2E 63 70 2E 72
000200:   75 2F 43 65 72 74 45 6E 72 6F 6C 6C 2F 76 6D 2D
000210:   74 65 73 74 2D 63 61 2E 63 70 2E 72 75 5F 74 65
000220:   73 74 2D 63 61 2E 63 72 74 30 4D 06 08 2B 06 01
000230:   05 05 07 30 02 86 41 66 69 6C 65 3A 2F 2F 5C 5C
000240:   76 6D 2D 74 65 73 74 2D 63 61 2E 63 70 2E 72 75
000250:   5C 43 65 72 74 45 6E 72 6F 6C 6C 5C 76 6D 2D 74
000260:   65 73 74 2D 63 61 2E 63 70 2E 72 75 5F 74 65 73
000270:   74 2D 63 61 2E 63 72 74 30 08 06 06 2A 85 03 02
000280:   02 03 03 41 00 93 30 50 11 50 42 80 3B 6F DD 1D
000290:   99 6A 75 0A C8 DA 2C 3E F2 28 47 5E D3 FB C7 9A
0002A0:   3E A1 C7 D5 80 AE 08 D0 81 F3 14 B4 88 09 BD 2C
0002B0:   D4 B5 8F A8 4C B2 B6 66 11 FD 6C 0A 84 BE 59 25
0002C0:   3D 18 87 CC 02
```

<-------

Формирование сообщения ServerHelloDone на стороне сервера:

```
msg_type:                 0E
length:                   000000
body:                     --
```

Итоговое сообщение ServerHelloDone:

```
000000:   0E 00 00 00
```

Формирование незащищенной записи с номером 2 на стороне сервера:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 0004
fragment:               0E000000
```

Итоговая запись с номером 2 на стороне сервера:

```
<-------  000000:   16 03 03 00 04 0E 00 00 00
```

Значение *PMS*, сгенерированное на стороне клиента:

```
000000:   A5 57 6C E7 92 4A 24 F5 81 13 80 8D BD 9E F8 56
000010:   F5 BD C3 B1 83 CE 5D AD CA 36 A5 3A A0 77 65 1D
```

Случайное значение k_{EPH}, выбранное на стороне клиента:

```
0xA5C77C7482373DE16CE4A6F73CCE7F78
  471493FF2C0709B8B706C9E8A25E6C1E
```

Эфемерный ключ Q_{EPH}:

```
x = 0x86D9192A6C85737E2AF5CC7C3761F3D9
      BE127378C218E27439E230B04C0F498D,
y = 0x71190E76B8E48FD9F172D9610A4E6CEC
      2D5E088CCDA71B245B40C636B8C1CE00
```

Хэш-значение *H* от конкатенации случайных строк клиента и сервера:

```
000000:   C3 EF 04 28 D4 B7 A1 F4 C5 02 5F 2E 65 DD 2B 2E
000010:   A5 83 AE EF DB 67 C7 F4 21 4A 6A 29 8E 99 E3 25
```

Генерация ключей экспорта. Значение *r*:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Алгоритм генерации ключей экспорта. Значение *UKM*:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Генерация ключей экспорта. Значение *seed*:

```
000000:   A5 83 AE EF DB 67 C7 F4
```

Генерация ключей экспорта. Значение K_{EXP}:

```
000000:   45 2C 59 66 54 DC 02 97 DF 5A C5 97 54 2F 99 04
000010:   49 5C D0 66 9A B9 CA 00 5D 92 E4 41 35 82 BE 76
```

Ключи экспорта $K_{MAC}^{Exp} \mid K_{ENC}^{Exp}$ для вычисления имитовставки и шифрования:

```
000000:   BA 46 01 4B 6F 47 60 38 45 52 3B A9 BB D5 A0 6F
000010:   15 37 54 C7 65 72 AE 81 C5 29 AB A4 D4 6E C5 27
000020:   FE 98 27 44 58 70 32 40 EB 45 F3 E2 49 EA 14 4C
000030:   F9 21 3B 3E 05 A1 D1 55 07 CE D1 F9 19 80 16 E8
```

Значение вектора инициализации *IV*:

```
000000:   21 4A 6A 29
```

Экспортное представление *PMSEXP* значения *PMS*:

```
000000:   25 36 55 6C CD AC 34 91 4F D1 15 4C 2A 9F 9E 5D
000010:   7F DE 77 43 50 FD 66 90 7A 20 21 A9 A1 DF 8C 98
000020:   2F 30 CF 2B E4 CF 91 AF
```

Формирование сообщения ClientKeyExchange на стороне клиента:

```
msg_type:               10
length:                 000097
```

```
body:
  exchange_keys:          30819404282536556CCDAC34914FD115
                          4C2A9F9E5D7FDE774350FD66907A2021
                          A9A1DF8C982F30CF2BE4CF91AF306830
                          2106082A85030701010101301506092A
                          85030701020101010106082A8503070101
                          02020343000408D490F4CB030E23974
                          E218C2787312BED9F361377CCCF52A7E
                          73856C2A19D98600CEC1B836C6405B24
                          1BA7CD8C085E2DEC6C4E0A61D972F1D9
                          8FE4B8760E1971
```

Итоговое сообщение ClientKeyExchange:

```
000000:   10 00 00 97 30 81 94 04 28 25 36 55 6C CD AC 34
000010:   91 4F D1 15 4C 2A 9F 9E 5D 7F DE 77 43 50 FD 66
000020:   90 7A 20 21 A9 A1 DF 8C 98 2F 30 CF 2B E4 CF 91
000030:   AF 30 68 30 21 06 08 2A 85 03 07 01 01 01 01 30
000040:   15 06 09 2A 85 03 07 01 02 01 01 01 06 08 2A 85
000050:   03 07 01 01 02 02 03 43 00 04 40 8D 49 0F 4C B0
000060:   30 E2 39 74 E2 18 C2 78 73 12 BE D9 F3 61 37 7C
000070:   CC F5 2A 7E 73 85 6C 2A 19 D9 86 00 CE C1 B8 36
000080:   C6 40 5B 24 1B A7 CD 8C 08 5E 2D EC 6C 4E 0A 61
000090:   D9 72 F1 D9 8F E4 B8 76 0E 19 71
```

Формирование незащищенной записи с номером 1 на стороне клиента

```
type:                     16
version:
  major:                  03
  minor:                  03
length:                   009B
fragment:                 10000097308194042825365 56CCDAC34
                          914FD1154C2A9F9E5D7FDE774350FD66
                          907A2021A9A1DF8C982F30CF2BE4CF91
                          AF3068302106082A8503070101010130
                          1506092A85030701020101010 6082A85
                          03070101020203430004408D490F4CB0
                          30E23974E218C2787312BED9F361377C
                          CCF52A7E73856C2A19D98600CEC1B836
                          C6405B241BA7CD8C085E2DEC6C4E0A61
                          D972F1D98FE4B8760E1971
```

Итоговая запись с номером 1 на стороне клиента:

```
000000:   16 03 03 00 9B 10 00 00 97 30 81 94 04 28 25 36
000010:   55 6C CD AC 34 91 4F D1 15 4C 2A 9F 9E 5D 7F DE
000020:   77 43 50 FD 66 90 7A 20 21 A9 A1 DF 8C 98 2F 30
000030:   CF 2B E4 CF 91 AF 30 68 30 21 06 08 2A 85 03 07
000040:   01 01 01 01 30 15 06 09 2A 85 03 07 01 02 01 01
000050:   01 06 08 2A 85 03 07 01 01 02 02 03 43 00 04 40
000060:   8D 49 0F 4C B0 30 E2 39 74 E2 18 C2 78 73 12 BE
000070:   D9 F3 61 37 7C CC F5 2A 7E 73 85 6C 2A 19 D9 86
000080:   00 CE C1 B8 36 C6 40 5B 24 1B A7 CD 8C 08 5E 2D
000090:   EC 6C 4E 0A 61 D9 72 F1 D9 8F E4 B8 76 0E 19 71
```

-------->

Извлечение экспортного представления *PMSEXP*:

```
000000:   25 36 55 6C CD AC 34 91 4F D1 15 4C 2A 9F 9E 5D
000010:   7F DE 77 43 50 FD 66 90 7A 20 21 A9 A1 DF 8C 98
000020:   2F 30 CF 2B E4 CF 91 AF
```

Хэш-значение *H* от конкатенации случайных строк клиента и сервера:

```
000000:   C3 EF 04 28 D4 B7 A1 F4 C5 02 5F 2E 65 DD 2B 2E
000010:   A5 83 AE EF DB 67 C7 F4 21 4A 6A 29 8E 99 E3 25
```

Генерация ключей экспорта. Значение *r*:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Генерация ключей экспорта. Значение *UKM*:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Генерация ключей экспорта. Значение *seed*:

```
000000:   A5 83 AE EF DB 67 C7 F4
```

Генерация ключей экспорта. Значение K_{EXP}:

```
000000:   45 2C 59 66 54 DC 02 97 DF 5A C5 97 54 2F 99 04
000010:   49 5C D0 66 9A B9 CA 00 5D 92 E4 41 35 82 BE 76
```

Ключи экспорта $K_{MAC}^{Exp} \mid K_{ENC}^{Exp}$ для вычисления имитовставки и шифрования:

```
000000:   BA 46 01 4B 6F 47 60 38 45 52 3B A9 BB D5 A0 6F
000010:   15 37 54 C7 65 72 AE 81 C5 29 AB A4 D4 6E C5 27
000020:   FE 98 27 44 58 70 32 40 EB 45 F3 E2 49 EA 14 4C
000030:   F9 21 3B 3E 05 A1 D1 55 07 CE D1 F9 19 80 16 E8
```

Значение вектора инициализации *IV*:

```
000000:   21 4A 6A 29
```

Извлеченный общий секрет *PMS*:

```
000000:   A5 57 6C E7 92 4A 24 F5 81 13 80 8D BD 9E F8 56
000010:   F5 BD C3 B1 83 CE 5D AD CA 36 A5 3A A0 77 65 1D
```

Хэш-значение $HASH(HM)$ конкатенации всех сообщений протокола Handshake для формирования *MS*:

```
000000:   61 FB 2C 5B 70 73 65 8F 94 14 ED 28 29 A0 F3 BA
000010:   96 42 27 0C C4 53 2E 33 FE 16 60 72 39 7C 62 CD
```

Общее секретное значение *MS*, выработанное на стороне клиента:

```
000000:   BC 56 49 C5 A8 C7 33 E8 E3 5C 06 7E 84 4F 35 6E
000010:   DA F3 6E BF 33 4E CF C8 EB FB 71 C4 74 3E 18 C7
000020:   D3 F7 B7 86 C2 98 BB 54 B6 C4 9C DC 1B E2 BA E1
```

Ключи вычисления кода аутентификации, ключи шифрования и векторы инициализации для каждого из состояний чтения и записи $K_{MAC,C}^{write} \mid K_{MAC,C}^{read} \mid K_{ENC,C}^{write} \mid K_{ENC,C}^{read} \mid IV_C^{write} \mid IV_C^{read}$:

```
000000:   85 F6 45 AD 9C EF C0 B0 C8 2B 42 1A 39 FD 2B AE
000010:   53 10 40 B1 96 8B AE DC 7A 88 0E F8 1F 27 7B F6
000020:   96 16 AB 12 72 39 DF 5D FC CB D4 08 B5 BD C6 42
000030:   8D 5A 3C 70 C1 1A 8F A1 A3 25 01 12 4E 73 11 78
000040:   C9 72 81 91 8E 3D 50 BC 41 97 1A F6 8F 38 37 BF
000050:   23 C7 0A E3 F2 2C BE F7 59 14 D6 57 E9 EA 01 DB
000060:   E9 FB DF EB EF 0C 29 F6 0B E4 6D 30 C0 3C 52 CB
000070:   0C 0F 6B 27 8E 7B 0A A9 5F 23 0A F8 E7 33 54 06
000080:   0D 40 AF 14 5A 5C 9E 93
```

Хэш-значение $HASH(HM)$ конкатенации всех сообщений протокола Handshake для формирования *MS*:

```
000000:   61 FB 2C 5B 70 73 65 8F 94 14 ED 28 29 A0 F3 BA
000010:   96 42 27 0C C4 53 2E 33 FE 16 60 72 39 7C 62 CD
```

Общее секретное значение *MS*, выработанное на стороне сервера:

```
000000:   BC 56 49 C5 A8 C7 33 E8 E3 5C 06 7E 84 4F 35 6E
000010:   DA F3 6E BF 33 4E CF C8 EB FB 71 C4 74 3E 18 C7
000020:   D3 F7 B7 86 C2 98 BB 54 B6 C4 9C DC 1B E2 BA E1
```

Ключи вычисления кода аутентификации, ключи шифрования и векторы инициализации для каждого из состояний записи и чтения $K^{read}_{MAC,S} \mid K^{write}_{MAC,S} \mid K^{read}_{ENC,S} \mid K^{write}_{ENC,S} \mid IV^{read}_{S} \mid IV^{write}_{S}$:

```
000000:   85 F6 45 AD 9C EF C0 B0 C8 2B 42 1A 39 FD 2B AE
000010:   53 10 40 B1 96 8B AE DC 7A 88 0E F8 1F 27 7B F6
000020:   96 16 AB 12 72 39 DF 5D FC CB D4 08 B5 BD C6 42
000030:   8D 5A 3C 70 C1 1A 8F A1 A3 25 01 12 4E 73 11 78
000040:   C9 72 81 91 8E 3D 50 BC 41 97 1A F6 8F 38 37 BF
000050:   23 C7 0A E3 F2 2C BE F7 59 14 D6 57 E9 EA 01 DB
000060:   E9 FB DF EB EF 0C 29 F6 0B E4 6D 30 C0 3C 52 CB
000070:   0C 0F 6B 27 8E 7B 0A A9 5F 23 0A F8 E7 33 54 06
000080:   0D 40 AF 14 5A 5C 9E 93
```

Формирование сообщения ChangeCipherSpec на стороне клиента:

```
type:                   01
```

Итоговое сообщение ChangeCipherSpec:

```
000000:   01
```

Формирование незащищенной записи с номером 2 на стороне клиента:

```
type:                   14
version:
  major:                03
  minor:                03
length:                 0001
fragment:               01
```

Итоговая запись с номером 2 на стороне клиента:

```
000000:   14 03 03 00 01 01
```

-------->

Хэш-значение $HASH(HM)$ конкатенации всех сообщений протокола Handshake для формирования *client _verify _data*:

```
000000:   61 FB 2C 5B 70 73 65 8F 94 14 ED 28 29 A0 F3 BA
000010:   96 42 27 0C C4 53 2E 33 FE 16 60 72 39 7C 62 CD
```

Данные *client _verify _data*:

```
000000:   7E 25 94 46 10 69 1B ED 2E 78 15 AE FA CE 93 45
000010:   9B DD D2 A6 B1 A3 3A C9 7C DD 69 D8 D8 F8 EA 33
```

Формирование сообщения Finished на стороне клиента:

```
type:                   14
length:                 000020
body:
  verify_data:          7E25944610691BED2E7815AEFACE9345
                        9BDDD2A6B1A33AC97CDD69D8D8F8EA33
```

Итоговое сообщение Finished:

```
000000:   14 00 00 20 7E 25 94 46 10 69 1B ED 2E 78 15 AE
000010:   FA CE 93 45 9B DD D2 A6 B1 A3 3A C9 7C DD 69 D8
000020:   D8 F8 EA 33
```

Формирование защищенной записи с номером 0 на стороне клиента:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 002C
fragment:               0CA6A44828011B82E436AD7B597EA2BD
                        8EBFA401AD9C18223E286285DD6C8125
                        C0C2EAAE9A5B2D0E40DDCA0F
```

Итоговая защищенная запись с номером 0 на стороне клиента:

```
000000:   16 03 03 00 2C 0C A6 A4 48 28 01 1B 82 E4 36 AD
000010:   7B 59 7E A2 BD 8E BF A4 01 AD 9C 18 22 3E 28 62
000020:   85 DD 6C 81 25 C0 C2 EA AE 9A 5B 2D 0E 40 DD CA
000030:   0F
```

-------->

Формирование сообщения ChangeCipherSpec на стороне сервера:

```
type:                   01
```

Итоговое сообщение ChangeCipherSpec:

```
000000:   01
```

Формирование незащищенной записи с номером 3 на стороне сервера:

```
type:                   14
version:
  major:                03
  minor:                03
length:                 0001
fragment:               01
```

Итоговая запись с номером 3 на стороне сервера:

<-------

```
000000:   14 03 03 00 01 01
```

Хэш-значение *HASH*(*HM*) конкатенации всех сообщений протокола Handshake для формирования *server _verify _data*:

```
000000:   CE 25 31 8E 4F A2 46 0D 4C A9 27 7E 6F 29 C5 25
000010:   52 0E 30 FA 0C 97 F2 75 06 D7 4D 9F A9 B4 3C 55
```

Данные *server _verify _data*:

```
000000:   2A 75 BE 8D B1 28 18 20 C3 E9 1C 3A CF B3 56 E5
000010:   38 BD C6 40 DA 0A 81 63 59 86 F3 D2 8C 39 15 21
```

Формирование сообщения Finished на стороне сервера:

```
type:                   14
length:                 000020
body:
  verify_data:          2A75BE8DB1281820C3E91C3ACFB356E5
                        38BDC640DA0A81635986F3D28C391521
```

Итоговое сообщение Finished:

```
000000:   14 00 00 20 2A 75 BE 8D B1 28 18 20 C3 E9 1C 3A
000010:   CF B3 56 E5 38 BD C6 40 DA 0A 81 63 59 86 F3 D2
000020:   8C 39 15 21
```

Формирование защищенной записи с номером 0 на стороне сервера:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 002C
fragment:               05DA23CF5F7FA6F635AE9FF2CF13FAA3
                        8EA7C66595E1F2A99F7E70E19B13B21C
                        D8F3B01526AFB670D22B73C9
```

Итоговая защищенная запись с номером 0 на стороне сервера:

<-------

```
000000:   16 03 03 00 2C 05 DA 23 CF 5F 7F A6 F6 35 AE 9F
000010:   F2 CF 13 FA A3 8E A7 C6 65 95 E1 F2 A9 9F 7E 70
000020:   E1 9B 13 B2 1C D8 F3 B0 15 26 AF B6 70 D2 2B 73
000030:   C9
```

Данные, передающиеся в первом сообщении протокола Application Data, посылаемом клиентом после завершения протокола Handshake:

```
000000:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Формирование защищенной записи с номером 1 на стороне клиента:

```
type:                   17
version:
  major:                03
  minor:                03
length:                 0028
fragment:               D44DF1A3077B030FDE90C001BFA344BB
                        D22F7F0F13B1B8114380646B6BEAE4F3
                        7D9899D476442753
```

Итоговая защищенная запись с номером 1 на стороне клиента:

```
000000:   17 03 03 00 28 D4 4D F1 A3 07 7B 03 0F DE 90 C0
000010:   01 BF A3 44 BB D2 2F 7F 0F 13 B1 B8 11 43 80 64   -------->
000020:   6B 6B EA E4 F3 7D 98 99 D4 76 44 27 53
```

Данные, передающиеся в первом сообщении протокола Application Data, посылаемом сервером после завершения протокола Handshake:

```
000000:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000010:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Формирование защищенной записи с номером 1 на стороне сервера:

```
type:                   17
version:
  major:                03
  minor:                03
length:                 0028
fragment:               42219CF2EB265D9C50CF23EEB21FA0B1
                        6A5DFBBDFF348BC55FF190B9B88B551C
                        04ECA7666B79FBFC
```

Итоговая защищенная запись с номером 1 на стороне сервера:

```
000000:   17 03 03 00 28 42 21 9C F2 EB 26 5D 9C 50 CF 23
000010:   EE B2 1F A0 B1 6A 5D FB BD FF 34 8B C5 5F F1 90
000020:   B9 B8 8B 55 1C 04 EC A7 66 6B 79 FB FC
```

<-------

Формирование оповещения close_notify на стороне клиента:

```
Alert:
  level:                01
  description:          00
```

Итоговое оповещение на стороне клиента:

```
000000:   01 00
```

Формирование защищенной записи с номером 2 на стороне клиента:

```
type:                   15
version:
  major:                03
  minor:                03
length:                 000A
fragment:               8FBAB978D7EABEECACC3
```

Итоговая защищенная запись с номером 2 на стороне клиента:

```
000000:   15 03 03 00 0A 8F BA B9 78 D7 EA BE EC AC C3   -------->
```

Формирование оповещения close_notify на стороне сервера:

```
Alert:
  level:                01
  description:          00
```

Итоговое оповещение на стороне клиента:

```
000000:   01 00
```

Формирование защищенной записи с номером 2 на стороне сервера:

```
type:                   15
version:
  major:                03
  minor:                03
length:                 000A
fragment:               777BEEF672FED88A6BE2
```

Итоговая защищенная запись с номером 2 на стороне сервера:

```
<-------  000000:   15 03 03 00 0A 77 7B EE F6 72 FE D8 8A 6B E2
```

B.2 TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC

В настоящем разделе приведен тестовый пример сценария работы протокола TLS 1.2 для криптонабора TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC, при котором используется полная схема работы протокола Handshake с двусторонней аутентификацией. Открытый и закрытый ключ сервера задаются следующим образом.

Идентификатор кривой сертификата сервера:
id-tc26-gost-3410-2012-512-paramSetC, «1.2.643.7.1.2.1.2.3».

Открытый ключ сервера Q_S:

```
x = 0xF14589DA479AD972C66563669B3FF580
      92E6A30A288BF447CD9FF6C3133E9724
      7A9706B267703C9B4E239F0D7C7E3310
      C22D2752B35BD2E4FD39B8F11DEB833A;
y = 0xF305E95B36502D4E60A1059FB20AB30B
      FC7C95727F3A2C04B1DFDDB53B0413F2
      99F2DFE66A5E1CCB4101A7A01D612BE6
      BD78E1E3B3D567EBB16ABE587A11F4EA.
```

Закрытый ключ сервера k_S:

```
0x12FD7A70067479A0F66C59F9A25534AD
  FBC7ABFD3CC72D79806F8B402601644B
  3005ED365A2D8989A8CCAE640D5FC08D
  D27DFBBFE137CF528E1AC6D445192E01.
```

Ключ проверки подписи и ключ подписи клиента задаются следующим образом.

Идентификатор кривой сертификата клиента:
id-tc26-gost-3410-2012-256-paramSetA, «1.2.643.7.1.2.1.1.1».

Ключ проверки подписи клиента Q_C:

```
x = 0x0F5DB18A9E15F324B778676025BFD7B5
      DF066566EABAA1C51CD879F87B0B4975;
y = 0x9EE5BBF18361F842D3F087DEC2943939
      E0FA2BFB4EDEC25A8D10ABB22C48F386.
```

Ключ подписи клиента k_C:

```
0x0918AD3F7D209ABF89F1E8505DA894CE
  E10DA09D32E72E815D9C0ADA30B5A103.
```

Предполагается, что соединение устанавливается впервые. Для удобства в рамках каждой записи передается только одно сообщение.

Формирование сообщения ClientHello на стороне клиента:

```
msg_type:                 01
length:                   000040
body:
  client_version:
    major:                03
    minor:                03
  random:                 933EA21EC3802A561550EC78D6ED51AC
                          2439D7E749C31BC3A3456165889684CA
  session_id:
    length:               00
    vector:               --
  cipher_suites:
    length:               0004
    vector:
      CipherSuite:        FF88
      CipherSuite:        FF89
  compression_methods:
    length:               01
    vector:
      CompressionMethod: 00
  extensions:
    length:               0013
    vector:
      Extension: /* signature_algorithms */
        extension_type:  000D
        extension_data:
          length:         0006
          vector:
            supported_signature_algorithms:
              length:     0004
              vector:
                /* Первая пара алгоритмов */
                hash:     EE
                signature:
                          EE
                /* Вторая пара алгоритмов */
                hash:     EF
                signature:
                          EF
      Extension: /* renegotiation_info */
        extension_type:  FF01
        extension_data:
          length:         0001
          vector:
            renegotiated_connection:
              length:     00
              vector:     --
      Extension: /* extended_master_secret */
        extension_type:  0017
        extension_data:
          length:         0000
          vector:         --
```

Итоговое сообщение ClientHello:

```
000000:   01 00 00 40 03 03 93 3E A2 1E C3 80 2A 56 15 50
000010:   EC 78 D6 ED 51 AC 24 39 D7 E7 49 C3 1B C3 A3 45
000020:   61 65 88 96 84 CA 00 00 04 FF 88 FF 89 01 00 00
000030:   13 00 0D 00 06 00 04 EE EE EF EF FF 01 00 01 00
000040:   00 17 00 00
```

Формирование незащищенной записи с номером 0 на стороне клиента:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 0044
fragment:               010000400303933EA21EC3802A561550
                        EC78D6ED51AC2439D7E749C31BC3A345
                        6165889684CA000004FF88FF89010000
                        13000D00060004EEEEEFEFFF01000100
                        00170000
```

Итоговая запись с номером 0 на стороне клиента:

```
000000:   16 03 03 00 44 01 00 00 40 03 03 93 3E A2 1E C3
000010:   80 2A 56 15 50 EC 78 D6 ED 51 AC 24 39 D7 E7 49
000020:   C3 1B C3 A3 45 61 65 88 96 84 CA 00 00 04 FF 88
000030:   FF 89 01 00 00 13 00 0D 00 06 00 04 EE EE EF EF
000040:   FF 01 00 01 00 00 17 00 00
```

-------->

Формирование сообщения ServerHello на стороне сервера:

```
msg_type:               02
length:                 000041
body:
  server_version:
    major:              03
    minor:              03
  random:               933EA21E49C31BC3A3456165889684CA
                        A5576CE7924A24F58113808DBD9EF856
  session_id:
    length:             10
    vector:             C3802A561550EC78D6ED51AC2439D7E7
  cipher_suite:
    CipherSuite:        FF89
  compression_method:
    CompressionMethod:  00
  extensions:
    length:             0009
    vector:
      Extension: /* renegotiation_info */
        extension_type: FF01
        extension_data:
          length:       0001
          vector:
            renegotiated_connection:
              length:   00
              vector:   --
      Extension: /* extended_master_secret */
        extension_type: 0017
        extension_data:
          length:       0000
          vector:       --
```

Итоговое сообщение ServerHello:

```
000000:   02 00 00 41 03 03 93 3E A2 1E 49 C3 1B C3 A3 45
000010:   61 65 88 96 84 CA A5 57 6C E7 92 4A 24 F5 81 13
000020:   80 8D BD 9E F8 56 10 C3 80 2A 56 15 50 EC 78 D6
000030:   ED 51 AC 24 39 D7 E7 FF 89 00 00 09 FF 01 00 01
000040:   00 00 17 00 00
```

Формирование незащищенной записи с номером 0 на стороне сервера:

```
type:                    16
version:
  major:                 03
  minor:                 03
length:                  0045
fragment:                020000410303933EA21E49C31BC3A345
                         6165889684CAA5576CE7924A24F58113
                         808DBD9EF85610C3802A561550EC78D6
                         ED51AC2439D7E7FF89000009FF010001
                         0000170000
```

Итоговая запись с номером 0 на стороне сервера:

<--------

```
000000:   16 03 03 00 45 02 00 00 41 03 03 93 3E A2 1E 49
000010:   C3 1B C3 A3 45 61 65 88 96 84 CA A5 57 6C E7 92
000020:   4A 24 F5 81 13 80 8D BD 9E F8 56 10 C3 80 2A 56
000030:   15 50 EC 78 D6 ED 51 AC 24 39 D7 E7 FF 89 00 00
000040:   09 FF 01 00 01 00 00 17 00 00
```

Формирование сообщения Certificate на стороне сервера:

```
msg_type:                0B
length:                  00024C
body:
  certificate_list:
    length:              000249
    vector:
      ASN.1Cert:
        length:          000246
        vector:          30820242308201AEA003020102020101
                         300A06082A850307010103033042312C
                         302A06092A864886F70D010901161D74
                                      . . .
                         371AF83C5BC58B366DFEFA7345D50317
                         867C177AC84AC07EE8612164629AB7BD
                         C48AA0F64A741FE7298E82C5BFCE8672
                         029F875391F7
```

Итоговое сообщение Certificate:

```
000000:   0B 00 02 4C 00 02 49 00 02 46 30 82 02 42 30 82
000010:   01 AE A0 03 02 01 02 02 01 01 30 0A 06 08 2A 85
000020:   03 07 01 01 03 03 30 42 31 2C 30 2A 06 09 2A 86
000030:   48 86 F7 0D 01 09 01 16 1D 74 6C 73 31 32 5F 73
000040:   65 72 76 65 72 35 31 32 43 40 63 72 79 70 74 6F
000050:   70 72 6F 2E 72 75 31 12 30 10 06 03 55 04 03 13
000060:   09 53 65 72 76 65 72 35 31 32 30 1E 17 0D 31 37
000070:   30 35 32 35 30 39 32 35 31 38 5A 17 0D 33 30 30
000080:   35 30 31 30 39 32 35 31 38 5A 30 42 31 2C 30 2A
000090:   06 09 2A 86 48 86 F7 0D 01 09 01 16 1D 74 6C 73
0000A0:   31 32 5F 73 65 72 76 65 72 35 31 32 43 40 63 72
0000B0:   79 70 74 6F 70 72 6F 2E 72 75 31 12 30 10 06 03
0000C0:   55 04 03 13 09 53 65 72 76 65 72 35 31 32 30 81
0000D0:   AA 30 21 06 08 2A 85 03 07 01 01 01 02 30 15 06
0000E0:   09 2A 85 03 07 01 02 01 02 03 06 08 2A 85 03 07
0000F0:   01 01 02 03 03 81 84 00 04 81 80 3A 83 EB 1D F1
000100:   B8 39 FD E4 D2 5B B3 52 27 2D C2 10 33 7E 7C 0D
000110:   9F 23 4E 9B 3C 70 67 B2 06 97 7A 24 97 3E 13 C3
000120:   F6 9F CD 47 F4 8B 28 0A A3 E6 92 80 F5 3F 9B 66
000130:   63 65 C6 72 D9 9A 47 DA 89 45 F1 EA F4 11 7A 58
000140:   BE 6A B1 EB 67 D5 B3 E3 E1 78 BD E6 2B 61 1D A0
000150:   A7 01 41 CB 1C 5E 6A E6 DF F2 99 F2 13 04 3B B5
```

```
000160:   DD DF B1 04 2C 3A 7F 72 95 7C FC 0B B3 0A B2 9F
000170:   05 A1 60 4E 2D 50 36 5B E9 05 F3 A3 43 30 41 30
000180:   1D 06 03 55 1D 0E 04 16 04 14 87 9C C6 5A 0F 4A
000190:   89 CB 4A 58 49 DF 05 61 56 9B AA DC 11 69 30 0B
0001A0:   06 03 55 1D 0F 04 04 03 02 03 28 30 13 06 03 55
0001B0:   1D 25 04 0C 30 0A 06 08 2B 06 01 05 05 07 03 01
0001C0:   30 0A 06 08 2A 85 03 07 01 01 03 03 03 81 81 00
0001D0:   35 BE 38 51 EC B6 E9 2D 32 40 01 81 0F 8C 89 03
0001E0:   52 42 F4 05 46 9F 4C 4E CB 05 02 7C 57 E2 71 52
0001F0:   12 AF D7 CD BB 0C ED 7A 8B 4D 33 42 CC 50 1A BD
000200:   99 99 75 A5 8A DE 0E 58 4F CA 35 F5 2E 45 58 B7
000210:   31 1D 49 D0 A0 51 32 79 F7 39 37 1A F8 3C 5B C5
000220:   8B 36 6D FE FA 73 45 D5 03 17 86 7C 17 7A C8 4A
000230:   C0 7E E8 61 21 64 62 9A B7 BD C4 8A A0 F6 4A 74
000240:   1F E7 29 8E 82 C5 BF CE 86 72 02 9F 87 53 91 F7
```

Формирование незащищенной записи с номером 1 на стороне сервера:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 0250
fragment:               0B00024C000249000246308202423082
                        01AEA003020102020101300A06082A85
                        03070101030330423 12C302A06092A86
                                        . . .
                        8B366DFEFA7345D50317867C177AC84A
                        C07EE8612164629AB7BDC48AA0F64A74
                        1FE7298E82C5BFCE8672029F875391F7
```

Итоговая запись с номером 1 на стороне сервера:

```
000000:   16 03 03 02 50 0B 00 02 4C 00 02 49 00 02 46 30
000010:   82 02 42 30 82 01 AE A0 03 02 01 02 02 01 01 30
000020:   0A 06 08 2A 85 03 07 01 01 03 03 30 42 31 2C 30
000030:   2A 06 09 2A 86 48 86 F7 0D 01 09 01 16 1D 74 6C
000040:   73 31 32 5F 73 65 72 76 65 72 35 31 32 43 40 63
000050:   72 79 70 74 6F 70 72 6F 2E 72 75 31 12 30 10 06
000060:   03 55 04 03 13 09 53 65 72 76 65 72 35 31 32 30
000070:   1E 17 0D 31 37 30 35 32 35 30 39 32 35 31 38 5A
000080:   17 0D 33 30 30 35 30 31 30 39 32 35 31 38 5A 30
000090:   42 31 2C 30 2A 06 09 2A 86 48 86 F7 0D 01 09 01
0000A0:   16 1D 74 6C 73 31 32 5F 73 65 72 76 65 72 35 31
0000B0:   32 43 40 63 72 79 70 74 6F 70 72 6F 2E 72 75 31
0000C0:   12 30 10 06 03 55 04 03 13 09 53 65 72 76 65 72
0000D0:   35 31 32 30 81 AA 30 21 06 08 2A 85 03 07 01 01
<-------
0000E0:   01 02 30 15 06 09 2A 85 03 07 01 02 01 02 03 06
0000F0:   08 2A 85 03 07 01 01 02 03 03 81 84 00 04 81 80
000100:   3A 83 EB 1D F1 B8 39 FD E4 D2 5B B3 52 27 2D C2
000110:   10 33 7E 7C 0D 9F 23 4E 9B 3C 70 67 B2 06 97 7A
000120:   24 97 3E 13 C3 F6 9F CD 47 F4 8B 28 0A A3 E6 92
000130:   80 F5 3F 9B 66 63 65 C6 72 D9 9A 47 DA 89 45 F1
000140:   EA F4 11 7A 58 BE 6A B1 EB 67 D5 B3 E3 E1 78 BD
000150:   E6 2B 61 1D A0 A7 01 41 CB 1C 5E 6A E6 DF F2 99
000160:   F2 13 04 3B B5 DD DF B1 04 2C 3A 7F 72 95 7C FC
000170:   0B B3 0A B2 9F 05 A1 60 4E 2D 50 36 5B E9 05 F3
000180:   A3 43 30 41 30 1D 06 03 55 1D 0E 04 16 04 14 87
000190:   9C C6 5A 0F 4A 89 CB 4A 58 49 DF 05 61 56 9B AA
0001A0:   DC 11 69 30 0B 06 03 55 1D 0F 04 04 03 02 03 28
0001B0:   30 13 06 03 55 1D 25 04 0C 30 0A 06 08 2B 06 01
0001C0:   05 05 07 03 01 30 0A 06 08 2A 85 03 07 01 01 03
0001D0:   03 03 81 81 00 35 BE 38 51 EC B6 E9 2D 32 40 01
```

```
0001E0:   81 0F 8C 89 03 52 42 F4 05 46 9F 4C 4E CB 05 02
0001F0:   7C 57 E2 71 52 12 AF D7 CD BB 0C ED 7A 8B 4D 33
000200:   42 CC 50 1A BD 99 99 75 A5 8A DE 0E 58 4F CA 35
000210:   F5 2E 45 58 B7 31 1D 49 D0 A0 51 32 79 F7 39 37
000220:   1A F8 3C 5B C5 8B 36 6D FE FA 73 45 D5 03 17 86
000230:   7C 17 7A C8 4A C0 7E E8 61 21 64 62 9A B7 BD C4
000240:   8A A0 F6 4A 74 1F E7 29 8E 82 C5 BF CE 86 72 02
000250:   9F 87 53 91 F7
```

Формирование сообщения CertificateRequest на стороне сервера:

```
msg_type:                0D
length:                  00000B
body:
  certificate_types:
    length:              02
    vector:
      /* gostr34102012_256*/
                         EE
      /* gostr34102012_512*/
                         EF
  supported_signature_algorithms:
    length:              0004
    vector:
     /* Первая пара алгоритмов */
     hash:               EE
     signature:          EE
     /* Вторая пара алгоритмов */
     hash:               EF
     signature:          EF
  certificate_authorities:
    length:              0000
    vector:              --
```

Итоговое сообщение CertificateRequest:

```
000000:   0D 00 00 0B 02 EE EF 00 04 EE EE EF EF 00 00
```

Формирование незащищенной записи с номером 2 на стороне сервера:

```
type:                    16
version:
  major:                 03
  minor:                 03
length:                  000F
fragment:                0D00000B02EEEF0004EEEEEFEF0000
```

Итоговая запись с номером 2 на стороне сервера:

```
<-------  000000:   16 03 03 00 0F 0D 00 00 0B 02 EE EF 00 04 EE EE
          000010:   EF EF 00 00
```

Формирование сообщения ServerHelloDone на стороне сервера:

```
msg_type:                0E
length:                  000000
body:                    --
```

Итоговое сообщение ServerHelloDone:

```
000000:   0E 00 00 00
```

Формирование незащищенной записи с номером 3 на стороне сервера:

```
type:                    16
version:
  major:                 03
```

```
                  minor:                  03
                length:                   0004
                fragment:                 0E000000
```

Итоговая запись с номером 3 на стороне сервера:

```
    <-------    000000:   16 03 03 00 04 0E 00 00 00
```

Формирование сообщения Certificate на стороне клиента:

```
msg_type:                 0B
length:                   0001EA
body:
  certificate_list:
    length:               0001E7
    vector:
      ASN.1Cert:
        length:           0001E4
        vector:           308201E03082018DA003020102020101
                          300A06082A85030701010302305312E
                          302C06092A864886F70D010901161F74
                                      . . .
                          C1CAB43AC01AFB0F3451BDC2DB188BBC
                          B77884251CDF6037BA830F4B31D5E96F
                          DC9BC1C95ABE658266C48402E070DE1F
                          292724E8
```

Итоговое сообщение Certificate:

```
000000:   0B 00 01 EA 00 01 E7 00 01 E4 30 82 01 E0 30 82
000010:   01 8D A0 03 02 01 02 02 01 01 30 0A 06 08 2A 85
000020:   03 07 01 01 03 02 30 53 31 2E 30 2C 06 09 2A 86
000030:   48 86 F7 0D 01 09 01 16 1F 74 6C 73 31 32 5F 63
000040:   6C 69 65 6E 74 32 35 36 41 5F 45 40 63 72 79 70
000050:   74 6F 70 72 6F 2E 72 75 31 21 30 1F 06 03 55 04
000060:   03 1E 18 00 43 00 6C 00 69 00 65 00 6E 00 74 00
000070:   32 00 35 00 36 00 41 00 5F 00 45 30 1E 17 0D 31
000080:   37 30 35 32 35 30 39 33 31 31 38 5A 17 0D 33 30
000090:   30 35 30 31 30 39 33 31 31 38 5A 30 53 31 2E 30
0000A0:   2C 06 09 2A 86 48 86 F7 0D 01 09 01 16 1F 74 6C
0000B0:   73 31 32 5F 63 6C 69 65 6E 74 32 35 36 41 5F 45
0000C0:   40 63 72 79 70 74 6F 70 72 6F 2E 72 75 31 21 30
0000D0:   1F 06 03 55 04 03 1E 18 00 43 00 6C 00 69 00 65
0000E0:   00 6E 00 74 00 32 00 35 00 36 00 41 00 5F 00 45
0000F0:   30 68 30 21 06 08 2A 85 03 07 01 01 01 01 30 15
000100:   06 09 2A 85 03 07 01 02 01 01 01 06 08 2A 85 03
000110:   07 01 01 02 02 03 43 00 04 40 75 49 0B 7B F8 79
000120:   D8 1C C5 A1 BA EA 66 65 06 DF B5 D7 BF 25 60 67
000130:   78 B7 24 F3 15 9E 8A B1 5D 0F 86 F3 48 2C B2 AB
000140:   10 8D 5A C2 DE 4E FB 2B FA E0 39 39 94 C2 DE 87
000150:   F0 D3 42 F8 61 83 F1 BB E5 9E A3 43 30 41 30 1D
000160:   06 03 55 1D 0E 04 16 04 14 74 49 1E 77 30 D3 42
000170:   A6 28 0E 72 A1 13 9D D9 90 8B FA F1 03 30 0B 06
000180:   03 55 1D 0F 04 04 03 02 07 80 30 13 06 03 55 1D
000190:   25 04 0C 30 0A 06 08 2B 06 01 05 05 07 03 02 30
0001A0:   0A 06 08 2A 85 03 07 01 01 03 02 03 41 00 1C 2D
0001B0:   35 22 B4 11 02 D6 20 1F 23 50 C1 CA B4 3A C0 1A
0001C0:   FB 0F 34 51 BD C2 DB 18 8B BC B7 78 84 25 1C DF
0001D0:   60 37 BA 83 0F 4B 31 D5 E9 6F DC 9B C1 C9 5A BE
0001E0:   65 82 66 C4 84 02 E0 70 DE 1F 29 27 24 E8
```

Формирование незащищенной записи с номером 1 на стороне клиента:

```
type:                     16
version:
```

```
  major:                  03
  minor:                  03
length:                   01EE
fragment:                 0B0001EA0001E70001E4308201E03082
                          018DA003020102020101300A06082A85
                          0307010103023053312E302C06092A86
                                        . . .
                          3522B41102D6201F2350C1CAB43AC01A
                          FB0F3451BDC2DB188BBCB77884251CDF
                          6037BA830F4B31D5E96FDC9BC1C95ABE
                          658266C48402E070DE1F292724E8
```

Итоговая запись с номером 1 на стороне клиента:

```
000000:   16 03 03 01 EE 0B 00 01 EA 00 01 E7 00 01 E4 30
000010:   82 01 E0 30 82 01 8D A0 03 02 01 02 02 01 01 30
000020:   0A 06 08 2A 85 03 07 01 01 03 02 30 53 31 2E 30
000030:   2C 06 09 2A 86 48 86 F7 0D 01 09 01 16 1F 74 6C
000040:   73 31 32 5F 63 6C 69 65 6E 74 32 35 36 41 5F 45
000050:   40 63 72 79 70 74 6F 70 72 6F 2E 72 75 31 21 30
000060:   1F 06 03 55 04 03 1E 18 00 43 00 6C 00 69 00 65
000070:   00 6E 00 74 00 32 00 35 00 36 00 41 00 5F 00 45
000080:   30 1E 17 0D 31 37 30 35 32 35 30 39 33 31 31 38
000090:   5A 17 0D 33 30 30 35 30 31 30 39 33 31 31 38 5A
0000A0:   30 53 31 2E 30 2C 06 09 2A 86 48 86 F7 0D 01 09
0000B0:   01 16 1F 74 6C 73 31 32 5F 63 6C 69 65 6E 74 32
0000C0:   35 36 41 5F 45 40 63 72 79 70 74 6F 70 72 6F 2E
0000D0:   72 75 31 21 30 1F 06 03 55 04 03 1E 18 00 43 00
0000E0:   6C 00 69 00 65 00 6E 00 74 00 32 00 35 00 36 00
0000F0:   41 00 5F 00 45 30 68 30 21 06 08 2A 85 03 07 01
000100:   01 01 01 30 15 06 09 2A 85 03 07 01 02 01 01 01
000110:   06 08 2A 85 03 07 01 01 02 02 03 43 00 04 40 75
000120:   49 0B 7B F8 79 D8 1C C5 A1 BA EA 66 65 06 DF B5
000130:   D7 BF 25 60 67 78 B7 24 F3 15 9E 8A B1 5D 0F 86
000140:   F3 48 2C B2 AB 10 8D 5A C2 DE 4E FB 2B FA E0 39
000150:   39 94 C2 DE 87 F0 D3 42 F8 61 83 F1 BB E5 9E A3
000160:   43 30 41 30 1D 06 03 55 1D 0E 04 16 04 14 74 49
000170:   1E 77 30 D3 42 A6 28 0E 72 A1 13 9D D9 90 8B FA
000180:   F1 03 30 0B 06 03 55 1D 0F 04 04 03 02 07 80 30
000190:   13 06 03 55 1D 25 04 0C 30 0A 06 08 2B 06 01 05
0001A0:   05 07 03 02 30 0A 06 08 2A 85 03 07 01 01 03 02
0001B0:   03 41 00 1C 2D 35 22 B4 11 02 D6 20 1F 23 50 C1
0001C0:   CA B4 3A C0 1A FB 0F 34 51 BD C2 DB 18 8B BC B7
0001D0:   78 84 25 1C DF 60 37 BA 83 0F 4B 31 D5 E9 6F DC
0001E0:   9B C1 C9 5A BE 65 82 66 C4 84 02 E0 70 DE 1F 29
0001F0:   27 24 E8
```

Значение *PMS*, сгенерированное на стороне клиента:

```
000000:   A5 57 6C E7 92 4A 24 F5 81 13 80 8D BD 9E F8 56
000010:   F5 BD C3 B1 83 CE 5D AD CA 36 A5 3A A0 77 65 1D
```

Случайное значение k_{EPH}, выбранное на стороне клиента:

```
0x550ACD11B66DD695AD18418FA7A2DC63
  6B7E29DCA24536AABC826EE3175BB1FA
  A5C77C7482373DE16CE4A6F73CCE7F78
  471493FF2C0709B8B706C9E8A25E6C1E
```

Эфемерный ключ Q_{EPH}:

```
x = 0xC941BE5193189B476D5A0334114A3E04
      BBE5B37C738AE40F150B334135288664
      FEBFC5622818894A07B1F7AD60E28480
      B4B637B90EA7D4BA980186B605D75BC6,
```

```
y = 0xA154F7B93E8148652011F4FD52C9A06A
      6471ADB28D0A949AE26BC786DE874153
      ABC00B35164F3214A8A83C00ECE27831
      B093528456234EFE766224FC2A7E9ABE
```

Хэш-значение *H* от конкатенации случайных строк клиента и сервера:

```
000000:   C3 EF 04 28 D4 B7 A1 F4 C5 02 5F 2E 65 DD 2B 2E
000010:   A5 83 AE EF DB 67 C7 F4 21 4A 6A 29 8E 99 E3 25
```

Генерация ключей экспорта. Значение *r*:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Алгоритм генерации ключей экспорта. Значение *UKM*:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Ключи экспорта $K_{MAC}^{Exp} \mid K_{ENC}^{Exp}$ для вычисления имитовставки и шифрования, являющиеся результатом работы функции VKO_{512}:

```
000000:   CA B8 4E 00 83 5A 63 8C AA 74 8C 1E 52 64 5E 1A
000010:   10 C8 F7 85 E2 40 D5 AF 29 E4 F6 73 FB 15 EA A0
000020:   C6 CD E4 A7 08 74 45 7D 3F 2A 1D 06 2C E2 97 58
000030:   37 F2 1D CA 63 CF 33 D7 30 B0 92 20 8D 99 05 1D
```

Значение вектора инициализации *IV*:

```
000000:   21 4A 6A 29 8E 99 E3 25
```

Экспортное представление *PMSEXP* значения *PMS*:

```
000000:   B8 EC 18 E3 33 23 4B 54 66 21 F3 B6 90 4A 5B 87
000010:   E4 AA 8B 8B BF 4C 95 13 AD E8 34 F5 25 58 91 94
000020:   58 D2 A0 AB C4 F1 6C 4B 52 A9 EE 0F B9 92 CD B8
```

Формирование сообщения ClientKeyExchange на стороне клиента:

```
msg_type:                10
length:                  0000E2
body:
  exchange_keys:         3081DF0430B8EC18E333234B546621F3
                         B6904A5B87E4AA8B8BBF4C9513ADE834
                         F52558919458D2A0ABC4F16C4B52A9EE
                                       . . .
                         93B03178E2EC003CA8A814324F16350B
                         C0AB534187DE86C76BE29A940A8DB2AD
                         71646AA0C952FDF411206548813EB9F7
                         54A1
```

Итоговое сообщение ClientKeyExchange:

```
000000:   10 00 00 E2 30 81 DF 04 30 B8 EC 18 E3 33 23 4B
000010:   54 66 21 F3 B6 90 4A 5B 87 E4 AA 8B 8B BF 4C 95
000020:   13 AD E8 34 F5 25 58 91 94 58 D2 A0 AB C4 F1 6C
000030:   4B 52 A9 EE 0F B9 92 CD B8 30 81 AA 30 21 06 08
000040:   2A 85 03 07 01 01 01 02 30 15 06 09 2A 85 03 07
000050:   01 02 01 02 03 06 08 2A 85 03 07 01 01 02 03 03
000060:   81 84 00 04 81 80 C6 5B D7 05 B6 86 01 98 BA D4
000070:   A7 0E B9 37 B6 B4 80 84 E2 60 AD F7 B1 07 4A 89
000080:   18 28 62 C5 BF FE 64 86 28 35 41 33 0B 15 0F E4
000090:   8A 73 7C B3 E5 BB 04 3E 4A 11 34 03 5A 6D 47 9B
0000A0:   18 93 51 BE 41 C9 BE 9A 7E 2A FC 24 62 76 FE 4E
0000B0:   23 56 84 52 93 B0 31 78 E2 EC 00 3C A8 A8 14 32
0000C0:   4F 16 35 0B C0 AB 53 41 87 DE 86 C7 6B E2 9A 94
0000D0:   0A 8D B2 AD 71 64 6A A0 C9 52 FD F4 11 20 65 48
0000E0:   81 3E B9 F7 54 A1
```

Формирование незащищенной записи с номером 2 на стороне клиента:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 00E6
fragment:               100000E23081DF0430B8EC18E333234B
                        546621F3B6904A5B87E4AA8B8BBF4C95
                        13ADE834F52558919458D2A0ABC4F16C
                                      . . .
                        2356845293B03178E2EC003CA8A81432
                        4F16350BC0AB534187DE86C76BE29A94
                        0A8DB2AD71646AA0C952FDF411206548
                        813EB9F754A1
```

Итоговая запись с номером 2 на стороне клиента:

```
000000:   16 03 03 00 E6 10 00 00 E2 30 81 DF 04 30 B8 EC
000010:   18 E3 33 23 4B 54 66 21 F3 B6 90 4A 5B 87 E4 AA
000020:   8B 8B BF 4C 95 13 AD E8 34 F5 25 58 91 94 58 D2
000030:   A0 AB C4 F1 6C 4B 52 A9 EE 0F B9 92 CD B8 30 81
000040:   AA 30 21 06 08 2A 85 03 07 01 01 01 02 30 15 06
000050:   09 2A 85 03 07 01 02 01 02 03 06 08 2A 85 03 07
000060:   01 01 02 03 03 81 84 00 04 81 80 C6 5B D7 05 B6
000070:   86 01 98 BA D4 A7 0E B9 37 B6 B4 80 84 E2 60 AD
000080:   F7 B1 07 4A 89 18 28 62 C5 BF FE 64 86 28 35 41
000090:   33 0B 15 0F E4 8A 73 7C B3 E5 BB 04 3E 4A 11 34
0000A0:   03 5A 6D 47 9B 18 93 51 BE 41 C9 BE 9A 7E 2A FC
0000B0:   24 62 76 FE 4E 23 56 84 52 93 B0 31 78 E2 EC 00
0000C0:   3C A8 A8 14 32 4F 16 35 0B C0 AB 53 41 87 DE 86
0000D0:   C7 6B E2 9A 94 0A 8D B2 AD 71 64 6A A0 C9 52 FD
0000E0:   F4 11 20 65 48 81 3E B9 F7 54 A1
```

-------->

Извлечение экспортного представления *PMSEXP*:

```
000000:   B8 EC 18 E3 33 23 4B 54 66 21 F3 B6 90 4A 5B 87
000010:   E4 AA 8B 8B BF 4C 95 13 AD E8 34 F5 25 58 91 94
000020:   58 D2 A0 AB C4 F1 6C 4B 52 A9 EE 0F B9 92 CD B8
```

Хэш-значение *H* от конкатенации случайных строк клиента и сервера:

```
000000:   C3 EF 04 28 D4 B7 A1 F4 C5 02 5F 2E 65 DD 2B 2E
000010:   A5 83 AE EF DB 67 C7 F4 21 4A 6A 29 8E 99 E3 25
```

Генерация ключей экспорта. Значение *r*:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Генерация ключей экспорта. Значение *UKM*:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Ключи экспорта $K_{MAC}^{Exp} | K_{ENC}^{Exp}$ для вычисления имитовставки и шифрования, являющиеся результатом работы функции VKO_{512}:

```
000000:   CA B8 4E 00 83 5A 63 8C AA 74 8C 1E 52 64 5E 1A
000010:   10 C8 F7 85 E2 40 D5 AF 29 E4 F6 73 FB 15 EA A0
000020:   C6 CD E4 A7 08 74 45 7D 3F 2A 1D 06 2C E2 97 58
000030:   37 F2 1D CA 63 CF 33 D7 30 B0 92 20 8D 99 05 1D
```

Значение вектора инициализации *IV*:

```
000000:   21 4A 6A 29 8E 99 E3 25
```

Извлеченный общий секрет *PMS*:

```
000000:   A5 57 6C E7 92 4A 24 F5 81 13 80 8D BD 9E F8 56
000010:   F5 BD C3 B1 83 CE 5D AD CA 36 A5 3A A0 77 65 1D
```

Случайное значение *k*, использующееся клиентом при формировании подписи:

```
0xD63962EEA268203E7C6B3F70BF8D4A36
  6458D7CC55B47928D85E7EAD25564E5F
```

Значение подписи $sgn_C = SIGN_{k_C}(HM)$:

```
000000:   F7 1F 43 62 45 5B C5 5B A8 9A 8F AF 01 82 88 EC
000010:   00 B3 27 17 48 2E 76 24 B2 57 D9 79 7C 8F F6 02
000020:   79 96 D8 46 27 60 9F F8 62 56 37 DF AE F4 A6 48
000030:   C4 A3 51 7C A6 5E 5B A3 79 4D C5 99 78 39 EF 1A
```

Формирование сообщения CertificateVerify на стороне клиента:

```
type:                   0F
length:                 000044
body:
  algorithm:
    hash:               EE
    signature:          EE
signature:
  length:               0040
  vector:               F71F4362455BC55BA89A8FAF018288EC
                        00B32717482E7624B257D9797C8FF602
                        7996D84627609FF8625637DFAEF4A648
                        C4A3517CA65E5BA3794DC5997839EF1A
```

Итоговое сообщение CertificateVerify:

```
000000:   0F 00 00 44 EE EE 00 40 F7 1F 43 62 45 5B C5 5B
000010:   A8 9A 8F AF 01 82 88 EC 00 B3 27 17 48 2E 76 24
000020:   B2 57 D9 79 7C 8F F6 02 79 96 D8 46 27 60 9F F8
000030:   62 56 37 DF AE F4 A6 48 C4 A3 51 7C A6 5E 5B A3
000040:   79 4D C5 99 78 39 EF 1A
```

Формирование незащищенной записи с номером 3 на стороне клиента:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 0048
fragment:               0F000044EEEE0040F71F4362455BC55B
                        A89A8FAF018288EC00B32717482E7624
                        B257D9797C8FF6027996D84627609FF8
                        625637DFAEF4A648C4A3517CA65E5BA3
                        794DC5997839EF1A
```

Итоговая запись с номером 3 на стороне клиента:

```
000000:   16 03 03 00 48 0F 00 00 44 EE EE 00 40 F7 1F 43
000010:   62 45 5B C5 5B A8 9A 8F AF 01 82 88 EC 00 B3 27
000020:   17 48 2E 76 24 B2 57 D9 79 7C 8F F6 02 79 96 D8   ------->
000030:   46 27 60 9F F8 62 56 37 DF AE F4 A6 48 C4 A3 51
000040:   7C A6 5E 5B A3 79 4D C5 99 78 39 EF 1A
```

Хэш-значение $HASH(HM)$ конкатенации всех сообщений протокола Handshake для формирования *MS*:

```
000000:   0A B0 3E 23 94 34 14 CE 60 19 F3 57 BA D0 2E 10
000010:   7D 17 FB C3 73 2B 58 CF 90 2A 59 E5 BB 50 22 CB
```

Общее секретное значение *MS*, выработанное на стороне клиента:

```
000000:   51 27 78 4A A7 99 04 2A BC 6B B9 A8 57 33 B1 24
000010:   78 BA 81 26 D1 C2 84 80 10 6C 67 82 78 89 02 06
000020:   E6 02 58 E1 AA 63 1E 72 C0 EC C9 A9 A4 E2 97 C3
```

Ключи вычисления кода аутентификации, ключи шифрования и векторы инициализации для каждого из состояний чтения и записи $K^{write}_{MAC,C} \mid K^{read}_{MAC,C} \mid K^{write}_{ENC,C} \mid K^{read}_{ENC,C} \mid IV^{write}_{C} \mid IV^{read}_{C}$:

```
000000:   5E B6 5A 74 00 52 7E C3 41 F0 00 76 8D 2B BE 42
000010:   94 EC 1E F5 C8 5A 0B E0 0A C6 1D 6E EA D0 41 33
000020:   BA 2D D8 EF C3 46 A7 E9 90 F7 F7 C6 DE BD F8 BE
000030:   42 78 51 74 A4 5F 15 32 30 7B 07 86 9A 38 8D 00
000040:   AC 71 F5 FF FE 6E 4A 31 E9 C0 06 73 5B C3 35 8A
000050:   D8 FD B2 9D 23 A6 DA 4E 36 E2 E9 50 A7 F5 CE 0D
000060:   1C 1D 0E 4E 55 B1 1B D9 6F 32 FD 22 AE FB 4F 34
000070:   7C 1A DC E2 1B 9B 98 49 9B E7 77 97 A3 1C 07 E1
000080:   6F A5 0F C3 1B C6 C5 40 F9 2D C3 AE 20 ED CF 27
```

Хэш-значение $HASH(HM)$ конкатенации всех сообщений протокола Handshake для формирования MS:

```
000000:   0A B0 3E 23 94 34 14 CE 60 19 F3 57 BA D0 2E 10
000010:   7D 17 FB C3 73 2B 58 CF 90 2A 59 E5 BB 50 22 CB
```

Общее секретное значение MS, выработанное на стороне сервера:

```
000000:   51 27 78 4A A7 99 04 2A BC 6B B9 A8 57 33 B1 24
000010:   78 BA 81 26 D1 C2 84 80 10 6C 67 82 78 89 02 06
000020:   E6 02 58 E1 AA 63 1E 72 C0 EC C9 A9 A4 E2 97 C3
```

Ключи вычисления кода аутентификации, ключи шифрования и векторы инициализации для каждого из состояний чтения и записи $K^{read}_{MAC,S} \mid K^{write}_{MAC,S} \mid K^{read}_{ENC,S} \mid K^{write}_{ENC,S} \mid IV^{read}_{S} \mid IV^{write}_{S}$:

```
000000:   5E B6 5A 74 00 52 7E C3 41 F0 00 76 8D 2B BE 42
000010:   94 EC 1E F5 C8 5A 0B E0 0A C6 1D 6E EA D0 41 33
000020:   BA 2D D8 EF C3 46 A7 E9 90 F7 F7 C6 DE BD F8 BE
000030:   42 78 51 74 A4 5F 15 32 30 7B 07 86 9A 38 8D 00
000040:   AC 71 F5 FF FE 6E 4A 31 E9 C0 06 73 5B C3 35 8A
000050:   D8 FD B2 9D 23 A6 DA 4E 36 E2 E9 50 A7 F5 CE 0D
000060:   1C 1D 0E 4E 55 B1 1B D9 6F 32 FD 22 AE FB 4F 34
000070:   7C 1A DC E2 1B 9B 98 49 9B E7 77 97 A3 1C 07 E1
000080:   6F A5 0F C3 1B C6 C5 40 F9 2D C3 AE 20 ED CF 27
```

Формирование сообщения ChangeCipherSpec на стороне клиента:

```
type:                   01
```

Итоговое сообщение ChangeCipherSpec:

```
000000:   01
```

Формирование незащищенной записи с номером 4 на стороне клиента:

```
type:                   14
version:
  major:                03
  minor:                03
length:                 0001
fragment:               01
```

Итоговая запись с номером 4 на стороне клиента:

```
000000:   14 03 03 00 01 01                                    -------->
```

Хэш-значение $HASH(HM)$ конкатенации всех сообщений протокола Handshake для формирования *client _verify _data*:

```
000000:   0A B0 3E 23 94 34 14 CE 60 19 F3 57 BA D0 2E 10
000010:   7D 17 FB C3 73 2B 58 CF 90 2A 59 E5 BB 50 22 CB
```

Данные *client _verify _data*:

```
000000:   6C 18 ED 88 6B 52 E2 DB BF 11 7E A3 24 F6 02 FE
000010:   11 86 08 B1 F1 98 FC 85 33 77 5D 90 E3 73 40 F1
```

Формирование сообщения Finished на стороне клиента:

```
type:                   14
length:                 000020
body:
  verify_data:          6C18ED886B52E2DBBF117EA324F602FE
                        118608B1F198FC8533775D90E37340F1
```

Итоговое сообщение Finished:

```
000000:   14 00 00 20 6C 18 ED 88 6B 52 E2 DB BF 11 7E A3
000010:   24 F6 02 FE 11 86 08 B1 F1 98 FC 85 33 77 5D 90
000020:   E3 73 40 F1
```

Формирование защищенной записи с номером 0 на стороне клиента:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 0034
fragment:               1DFC377D25D1CCD22A61D4C35424478F
                        64084B0997C3D6129CBBBA9C12D24A93
                        3883E78AC7C800B16661FB8147BF1690
                        FE2A94F4
```

Итоговая защищенная запись с номером 0 на стороне клиента:

```
000000:   16 03 03 00 34 1D FC 37 7D 25 D1 CC D2 2A 61 D4
000010:   C3 54 24 47 8F 64 08 4B 09 97 C3 D6 12 9C BB BA
000020:   9C 12 D2 4A 93 38 83 E7 8A C7 C8 00 B1 66 61 FB
000030:   81 47 BF 16 90 FE 2A 94 F4
```

-------->

Формирование сообщения ChangeCipherSpec на стороне сервера:

```
type:                   01
```

Итоговое сообщение ChangeCipherSpec:

```
000000:   01
```

Формирование незащищенной записи с номером 4 на стороне сервера:

```
type:                   14
version:
  major:                03
  minor:                03
length:                 0001
fragment:               01
```

Итоговая запись с номером 4 на стороне сервера:

<-------

```
000000:   14 03 03 00 01 01
```

Хэш-значение $HASH(HM)$ конкатенации всех сообщений протокола Handshake для формирования *server _verify _data*:

```
000000:   F1 37 16 1D D1 21 89 6A 39 13 B7 57 BA CC AD D5
000010:   9C 52 C5 A8 D4 45 E9 A9 45 00 9B 02 AA B1 3B 23
```

Данные *server _verify _data*:

```
000000:   11 0E 66 E6 EA 34 13 20 6C 73 A9 F4 36 EA 7F F5
000010:   0C B5 62 7C C1 62 9F 23 A7 34 EF B5 55 4C 71 A3
```

Формирование сообщения Finished на стороне сервера:

```
type:                   14
length:                 000020
body:
  verify_data:          110E66E6EA3413206C73A9F436EA7FF5
                        0CB5627CC1629F23A734EFB5554C71A3
```

Итоговое сообщение Finished:

```
000000:   14 00 00 20 11 0E 66 E6 EA 34 13 20 6C 73 A9 F4
000010:   36 EA 7F F5 0C B5 62 7C C1 62 9F 23 A7 34 EF B5
000020:   55 4C 71 A3
```

Формирование защищенной записи с номером 0 на стороне сервера:

```
type:                   16
version:
  major:                03
  minor:                03
length:                 0034
fragment:               7C9F32F58190651FC2E2B11B5E8C5AEC
                        5FD36BA46206AAD25BCB44266B4C3249
                        B6FA5A6B528420D076BAA3D2521578B3
                        43611361
```

Итоговая защищенная запись с номером 0 на стороне сервера:

```
000000:   16 03 03 00 34 7C 9F 32 F5 81 90 65 1F C2 E2 B1
000010:   1B 5E 8C 5A EC 5F D3 6B A4 62 06 AA D2 5B CB 44
000020:   26 6B 4C 32 49 B6 FA 5A 6B 52 84 20 D0 76 BA A3
000030:   D2 52 15 78 B3 43 61 13 61
```

<-------

Данные, передающиеся в первом сообщении протокола Application Data, посылаемом клиентом после завершения протокола Handshake:

```
000000:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Формирование защищенной записи с номером 1 на стороне клиента:

```
type:                   17
version:
  major:                03
  minor:                03
length:                 0030
fragment:               EE18002EC4B1CB28405E36DEDE2CCE8D
                        11F2B2A939A10D94BDD82633C74FAFE3
                        2A02FDBB3514AEDB05D500029ABFD169
```

Итоговая защищенная запись с номером 1 на стороне клиента:

```
000000:   17 03 03 00 30 EE 18 00 2E C4 B1 CB 28 40 5E 36
000010:   DE DE 2C CE 8D 11 F2 B2 A9 39 A1 0D 94 BD D8 26
000020:   33 C7 4F AF E3 2A 02 FD BB 35 14 AE DB 05 D5 00
000030:   02 9A BF D1 69
```

------->

Данные, передающиеся в первом сообщении протокола Application Data, посылаемом сервером после завершения протокола Handshake:

```
000000:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000010:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Формирование защищенной записи с номером 1 на стороне сервера:

```
type:                   17
version:
  major:                03
  minor:                03
length:                 0030
fragment:               BAEAAE34690007D3EA0D888C8BB94D3B
                        5AD6166EF672347AAE23C6DC20B2C817
                        9B45B5C2D96A3C7A293BDB0D4936EE5C
```

Итоговая защищенная запись с номером 1 на стороне сервера:

```
000000:   17 03 03 00 30 BA EA AE 34 69 00 07 D3 EA 0D 88
000010:   8C 8B B9 4D 3B 5A D6 16 6E F6 72 34 7A AE 23 C6
```

<-------

```
000020:   DC 20 B2 C8 17 9B 45 B5 C2 D9 6A 3C 7A 29 3B DB
000030:   0D 49 36 EE 5C
```

Формирование оповещения close_notify на стороне клиента:

```
Alert:
  level:                01
  description:          00
```

Итоговое оповещение на стороне клиента:

```
000000:   01 00
```

Формирование защищенной записи с номером 2 на стороне клиента:

```
type:                   15
version:
  major:                03
  minor:                03
length:                 0012
fragment:               C2AB4BB5C1558F69033B81935D190AAD
                        B7F1
```

Итоговая защищенная запись с номером 2 на стороне клиента:

```
000000:   15 03 03 00 12 C2 AB 4B B5 C1 55 8F 69 03 3B 81
000010:   93 5D 19 0A AD B7 F1
```

-------->

Формирование оповещения close_notify на стороне сервера:

```
Alert:
  level:                01
  description:          00
```

Итоговое оповещение на стороне сервера:

```
000000:   01 00
```

Формирование защищенной записи с номером 2 на стороне сервера:

```
type:                   15
version:
  major:                03
  minor:                03
length:                 0012
fragment:               E838B24A21759CF05B841848088C1555
                        320A
```

Итоговая защищенная запись с номером 2 на стороне сервера:

<--------

```
000000:   15 03 03 00 12 E8 38 B2 4A 21 75 9C F0 5B 84 18
000010:   48 08 8C 15 55 32 0A
```

Библиография

[1] IETF RFC 5246 T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, IETF RFC 5246, August 2008

[2] Техническая спецификация. Информационная технология. Криптографическая защита информации. Использование алгоритмов ГОСТ Р 34.10, ГОСТ Р 34.11 в профиле сертификата и списке отзыва сертификатов (CRL) инфраструктуры открытых ключей X.509 (утверждена решением заседания Технического комитета по стандартизации «Криптографическая защита информации»; Протокол № 13 от 24 апреля 2014 г.)

[3] IETF RFC 7627 K. Bhargavan, A. Delignat-Lavaud, A. Pironti, A. Langley, M. Ray, Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension, IETF 7627, September 2015

[4] IETF RFC 5746 E. Rescorla, M. Ray, S. Dispensa, N. Oskov, Transport Layer Security (TLS) Renegotiation Indication Extension, IETF 5746, February 2010

УДК 681.3.06:006.354 ОКС 35. 040 ОКСТУ 5002 П85

Ключевые слова: криптографические протоколы, аутентификация, пароль, ключ

БЗ 9—2018/13

Редактор *Н.А. Аргунова*
Технический редактор *В.Н. Прусакова*
Корректор *Е.Р. Ароян*
Компьютерная верстка *Ю.В. Поповой*

Сдано в набор 23.08.2018. Подписано в печать 10.09.2018. Формат 60 × 84 1/8. Гарнитура Ариал.
Усл. печ. л. 7,44. Уч.-изд. л. 6,73.

Подготовлено на основе электронной версии, предоставленной разработчиком рекомендаций

ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.
gost.e-ecolog.ru gost.e-ecolog.ru

Создано в единичном исполнении ФГУП «СТАНДАРТИНФОРМ»
для комплектования Федерального информационного фонда стандартов,
117418 Москва, Нахимовский пр-т, д. 31, к. 2. gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru