ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Р 1323565.1.032—2020

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Использование российских криптографических механизмов для реализации обмена данными по протоколу DLMS

Издание официальное

Москва
Стандартинформ
2020

Предисловие

1 РАЗРАБОТАНЫ Открытым акционерным обществом «Информационные технологии и коммуникационные системы» (ОАО «ИнфоТеКС»)

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 27 октября 2020 г. № 941-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 (www.gost.ru)

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 Криптографические наборы 5
5 Ключи 6
5.1 Симметричные ключи 6
5.2 Асимметричные ключи 7
6 Вспомогательные величины 8
6.1 Байт управления безопасностью 8
6.2 Идентификатор стороны 8
6.3 Счетчик вызовов 8
6.4 Сертификат открытого ключа 9
7 Описание алгоритмов 9
7.1 Шифрование и имитозащита 10
7.2 Передача ключа 11
7.3 Защита с помощью электронной цифровой подписи 13
7.4 Согласование ключа 14
7.5 Механизмы аутентификации HLS 22
Приложение А (справочное) Контрольные примеры 26
Библиография 35

Введение

В настоящих рекомендациях определяется порядок использования российских криптографических алгоритмов в протоколе DLMS, предназначенном для защищенного взаимодействия между системами сбора данных и измерительными устройствами. Протокол DLMS может быть использован для обеспечения конфиденциальности и целостности данных, а также для аутентификации источника данных. Протокол DLMS специфицируется в DLMS/COSEM Architecture and Protocols [1] и COSEM Interface Classes and OBIS Object Identification System [2].

Необходимость разработки настоящих рекомендаций вызвана потребностью в обеспечении совместимости реализаций протокола DLMS с использованием российских криптографических стандартов.

Примечание — Настоящие рекомендации дополнены приложением А.

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Использование российских криптографических механизмов для реализации обмена данными по протоколу DLMS

Information technology. Cryptographic data security. Usage of Russian cryptographic mechanisms to implement data exchange in the DLMS protocol

Дата введения — 2021—04—01

1 Область применения

Протокол DLMS рекомендуется применять при обмене данными между системами сбора данных и измерительными устройствами. Обмен данными основан на модели клиент/сервер, где системы сбора данных играют роль клиента, а измерительные устройства играют роль сервера.

2 Нормативные ссылки

В настоящих рекомендациях использованы нормативные ссылки на следующие стандарты:

ГОСТ 34.10—2018 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи

ГОСТ 34.11—2018 Информационная технология. Криптографическая защита информации. Функция хэширования

ГОСТ 34.12—2018 Информационная технология. Криптографическая защита информации. Блочные шифры

ГОСТ 34.13—2018 Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров

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

Р 1323565.1.023—2018 Информационная технология. Криптографическая защита информации. Использование алгоритмов ГОСТ Р 34.10—2012, ГОСТ Р 34.11—2012 в сертификате, списке аннулированных сертификатов (CRL) и запросе на сертификат PKCS #10 инфраструктуры открытых ключей X.509

Р 1323565.1.024—2019 Информационная технология. Криптографическая защита информации. Параметры эллиптических кривых для криптографических алгоритмов и протоколов

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 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 документ, 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

Издание официальное

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 применен следующий термин с соответствующим определением:

3.1.1 **криптографический набор:** Совокупность криптографических алгоритмов и параметров, используемых в протоколе DLMS.

3.2 Обозначения

В настоящих рекомендациях использованы следующие обозначения:

V^* — множество всех двоичных строк конечной длины, включая пустую строку;

$|x|$ — длина (число компонент) строки $x \in V^*$ (если x — пустая строка, то $|x| = 0$);

V_s — множество всех двоичных строк длины s, где s — целое неотрицательное число; нумерация подстрок и компонент строки осуществляется справа налево, начиная с нуля;

$x||y$ — конкатенация двоичных строк $x, y \in V^*$, то есть строка из $V_{|x|+|y|}$, в которой подстрока с большими номерами компонент из $V_{|x|}$ совпадает со строкой x, а подстрока с меньшими номерами компонент из $V_{|y|}$ совпадает со строкой y;

0^s — двоичная строка, состоящая из s нулей;

$\mathbb{Z}_s$ — кольцо вычетов по модулю s;

$LSB_s : V^* \setminus \bigcup_{i=0}^{s-1} V_i \to V_s$ — отображение, ставящее в соответствие строке $z_{m-1}||\ldots||z_1||z_0$, $m \geq s$ строку $z_{s-1}||\ldots||z_1||z_0$, $z_i \in V_1$, $i = 0, 1, \ldots, m-1$;

$MSB_s : V^* \setminus \bigcup_{i=0}^{s-1} V_i \to V_s$ — отображение, ставящее в соответствие строке $z_{m-1}||\ldots||z_1||z_0$, $m \geq s$ строку $z_{m-1}||\ldots||z_{m-s+1}||z_{m-s}$, $z_i \in V_1$, $i = 0, 1, \ldots, m-1$;

$Vec_s: \mathbb{Z}_{2^s} \to V_s$ — биективное отображение, сопоставляющее элементу кольца $\mathbb{Z}_{2^s}$ его двоичное представление, то есть для любого элемента $z \in \mathbb{Z}_{2^s}$, представленного в виде $z = z_0 + 2 \cdot z_1 + \ldots + 2^{s-1} \cdot z_{s-1}$, где $z_i \in \{0,1\}$, $i = 0,1,\ldots,s-1$, выполнено равенство $Vec_s(z) = z_{s-1}||\ldots||z_1||z_0$;

$Int_s: V_s \to \mathbb{Z}_{2^s}$ — отображение, обратное к отображению Vec_s, то есть $Int_s = Vec_s^{-1}$;

E — группа точек эллиптической кривой, задаваемая набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019;

p — характеристика простого поля, над которым определяется эллиптическая кривая E;

m — порядок группы точек эллиптической кривой E;

q — порядок циклической подгруппы группы точек эллиптической кривой E;

$\mathcal{O}$ — нулевая точка эллиптической кривой E;

P — точка эллиптической кривой E, являющаяся порождающим элементом циклической подгруппы порядка q, задаваемая набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019;

$\pi_x: E \to V_{256}$ — отображение, сопоставляющее точке эллиптической кривой E двоичное представление первой координаты этой точки в форме Вейерштрасса, то есть для любой точки $Q \in E$, представленной в форме Вейерштрасса $Q = (x_Q, y_Q)$, где $x_Q, y_Q \in \mathbb{Z}_p \subset \mathbb{Z}_{2^{256}}$, выполнено равенство $\pi_x(Q) = Vec_{256}(x_Q)$;

$\pi_y: E \to V_{256}$ — отображение, сопоставляющее точке эллиптической кривой E двоичное представление второй координаты этой точки в форме Вейерштрасса, то есть для любой точки $Q \in E$, представленной в форме Вейерштрасса $Q = (x_Q, y_Q)$, где $x_Q, y_Q \in \mathbb{Z}_p \subset \mathbb{Z}_{2^{256}}$, выполнено равенство $\pi_y(Q) = Vec_{256}(y_Q)$;

$HASH_{256}(M)$ — алгоритм вычисления хэш-функции с длиной хэш-кода 256 бит от данных M, определенный в ГОСТ 34.11—2018;

$KDFTREE(K, label, seed, R)$ — алгоритм диверсификации ключа KDF_TREE_GOSTR3411_2012_256, определенный в Р 50.1.113—2016, где K — ключ диверсификации, *label, seed* — параметры, задаваемые протоколом, $R \in \{1,2,3,4\}$;

$KUZN\_CMAC(K, M)$ — алгоритм шифрования ГОСТ 34.12—2018 («Кузнечик») в режиме выработки имитовставки согласно ГОСТ 34.13—2018, где K — ключ, M — данные. Значение входного параметра режима выработки имитовставки $s = 96$;

$KUZN\_CTR(K, IV, P)$ — алгоритм зашифрования ГОСТ 34.12—2018 («Кузнечик») в режиме работы, аналогичном режиму гаммирования согласно ГОСТ 34.13—2018, где K — ключ, IV — синхропосылка, P — открытые данные, с теми отличиями, что $IV \in V_{96}$ и начальное значение счетчика режима гаммирования вычисляется как $CTR_1 = IV||0^{32}$. Значение входного параметра режима гаммирования $s = 128$;

$KUZN\_KEXP(K, K_M, K_E, IV)$ — алгоритм экспорта ключа, аналогичный алгоритму экспорта ключа KExp15, определенному в Р 1323565.1.017—2018, на основе алгоритма шифрования ГОСТ 34.12—2018 («Кузнечик»), где K — передаваемый ключ, K_M — мастер-ключ имитозащиты, K_E — мастер-ключ шифрования, IV — синхропосылка, с теми отличиями, что $IV \in V_{96}$ и начальное значение счетчика режима гаммирования вычисляется как $CTR_1 = IV||0^{32}$;

$KUZN\_KIMP(ExpK, K_M, K_E, IV)$ — алгоритм импорта ключа, аналогичный алгоритму импорта ключа KImp15, определенному в Р 1323565.1.017—2018, на основе алгоритма шифрования ГОСТ 34.12—2018 («Кузнечик»), где $ExpK$ — экспортное представление передаваемого ключа, K_M — мастер-ключ имитозащиты, K_E — мастер-ключ шифрования, IV — синхропосылка, с теми отличиями, что $IV \in V_{96}$ и начальное значение счетчика режима гаммирования вычисляется как $CTR_1 = IV||0^{32}$;

$SIGN_{256}(d, M)$ — алгоритм формирования цифровой подписи согласно ГОСТ 34.10—2018 на основе эллиптической кривой E, где d — ключ подписи, M — подписываемые данные;

$VKO_{256}(d, Q, UKM)$ — алгоритм выработки ключа обмена VKO_GOST3410_2012_256, определенный в Р 50.1.113—2016, на основе эллиптической кривой E, где d — закрытый ключ вычисляющей стороны, Q — открытый ключ противоположной стороны, UKM — число (в формульных обозначениях Р 50.1.113—2016 вместо открытого ключа Q используется соответствующий ему закрытый ключ);

A-Associate — сервис ACSE, использующийся при установлении AA;

AA — Application Association: ассоциация приложений;

AARE — A-Associate Response: APDU, использующийся ACSE и предназначенный для осуществления ответа при использовании сервиса A-Associate;

AARQ	—	A-Associate Request: APDU, использующийся ACSE и предназначенный для осуществления запроса при использовании сервиса A-Associate;
ACSE	—	Association Control Service Element: элемент службы управления ассоциациями;
APDU	—	Application layer Protocol Data Unit: блок данных протокола уровня приложений;
COSEM	—	Companion Specification for Energy Metering: сопутствующая спецификация для учета электроэнергии;
DLMS	—	Device Language Message Specification: спецификация языка сообщений между устройствами;
HLS	—	High Level Security: высокий уровень безопасности;
xDLMS APDU	—	Extended DLMS APDU: APDU, использующийся xDLMS ASE;
xDLMS ASE	—	Extended DLMS Application Service Element: элемент службы приложений расширенной DLMS;
xDLMS InitiateRequest APDU	—	xDLMS APDU, применяющийся при установлении AA;
Data protection	—	объект COSEM, применяющийся при защите данных COSEM;
Security setup	—	объект COSEM, содержащий информацию об используемом криптографическом наборе и применяемой политике безопасности;
general-ciphering	—	способ защиты xDLMS APDU: шифрование общего вида;
general-signing	—	способ защиты xDLMS APDU: подпись общего вида;
general-ded-ciphering	—	способ защиты xDLMS APDU: специализированное шифрование общего вида;
general-glo-ciphering	—	способ защиты xDLMS APDU: глобальное шифрование общего вида;
service-specific dedicated ciphering	—	способ защиты xDLMS APDU: специализированное шифрование, зависящее от сервиса;
service-specific global ciphering	—	способ защиты xDLMS APDU: глобальное шифрование, зависящее от сервиса;
change_HLS_secret	—	метод объекта ассоциации COSEM, описанный в [2] и предназначенный для изменения значения общего секрета двух сторон при использовании механизмов аутентификации HLS (аутентификации высокого уровня безопасности);
key_agreement	—	метод объекта Security setup, описанный в [2] и предназначенный для пересылки сеансового открытого ключа и сопутствующей информации при согласовании ключа;
key_transfer	—	метод объекта Security setup, описанный в [2] и предназначенный для пересылки передаваемого ключа и сопутствующей информации при передаче ключа;
generate_key_pair	—	метод объекта Security setup, описанный в [2] и предназначенный для генерации ключевой пары сервера;
generate_certificate_request	—	метод объекта Security setup, описанный в [2] и предназначенный для создания сервером запроса на сертификат открытого ключа;
import_certificate	—	метод объекта Security setup, описанный в [2] и предназначенный для пересылки сертификата открытого ключа по направлению к серверу;
export_certificate	—	метод объекта Security setup, описанный в [2] и предназначенный для пересылки сертификата открытого ключа по направлению от сервера;
remove_certificate	—	метод объекта Security setup, описанный в [2] и предназначенный для удаления сертификата открытого ключа, хранящегося на сервере;

reply_to_HLS_authentication — метод объекта ассоциации COSEM, описанный в [2] и предназначенный для пересылки ответа на запрос при использовании механизмов аутентификации HLS (аутентификации высокого уровня безопасности).

4 Криптографические наборы

В настоящих рекомендациях специфицируется два криптографических набора для использования в протоколе DLMS. Описание криптографических наборов приведено в таблице 1.

Таблица 1 — Описание криптографических наборов

Параметры и алгоритмы криптографического набора	Криптографический набор 8	Криптографический набор 9
Идентификатор криптографического набора (Security Suite Id)	8	9
Название криптографического набора (Security Suite name)	KUZN-CTR-CMAC	VKO-256-GOST34102018-256-KUZN-CTR-CMAC-GOST34112018-256
Шифрование и имитозащита (Authenticated encryption)	Блочный шифр ГОСТ 34.12—2018 («Кузнечик») в режимах CTR* и CMAC, согласно ГОСТ 34.13—2018	Блочный шифр ГОСТ 34.12—2018 («Кузнечик») в режимах CTR* и CMAC, согласно ГОСТ 34.13—2018
Электронная цифровая подпись (Digital signature)	—	ГОСТ 34.10—2018 с длиной закрытого ключа 256 бит, с использованием эллиптической кривой, задаваемой набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019
Согласование ключа (Key agreement)	—	Алгоритмы согласования, основанные на VKO_GOSTR3410_2012_256, определенном в Р 50.1.113—2016, с использованием эллиптической кривой, задаваемой набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019
Хэш-функция (Hash)	—	ГОСТ 34.11—2018 с длиной хэш-кода 256 бит
Передача ключа (Key transport)	KExp15, KImp15**, определенные в Р 1323565.1.017—2018, на основе блочного шифра ГОСТ 34.12—2018 («Кузнечик»)	KExp15, KImp15**, определенные в Р 1323565.1.017—2018, на основе блочного шифра ГОСТ 34.12—2018 («Кузнечик»)

* Используемый в криптографических наборах 8 и 9 режим шифрования аналогичен режиму гаммирования согласно ГОСТ 34.13—2018 с теми отличиями, что длина используемой синхропосылки IV равна 96 бит и начальное значение счетчика режима гаммирования вычисляется как $CTR_1 = IV||0^{32}$.

** Используемые в криптографических наборах 8 и 9 алгоритмы экспорта и импорта ключа аналогичны алгоритмам KExp15, KImp15, определенным в Р 1323565.1.017—2018, на основе блочного шифра ГОСТ 34.12—2018 («Кузнечик»), с теми отличиями, что длина используемой синхропосылки IV равна 96 бит и начальное значение счетчика режима гаммирования вычисляется как $CTR_1 = IV||0^{32}$.

Использование криптографического набора 8 позволяет осуществлять:
- шифрование и имитозащиту данных (см. 7.1);
- передачу ключа (см. 7.2).

Использование криптографического набора 9 позволяет осуществлять:
- шифрование и имитозащиту данных (см. 7.1);
- передачу ключа (см. 7.2);
- защиту данных с помощью электронной цифровой подписи (см. 7.3);
- согласование ключа (см. 7.4).

При использовании в процессе установления АА механизмов аутентификации HLS (см. 7.5) допускается применение:

- механизма аутентификации «HLS CMAC» для согласования криптографического набора 8;
- механизмов аутентификации «HLS GOST34112018-256» и «HLS GOST34102018-256» для согласования криптографического набора 9.

5 Ключи

5.1 Симметричные ключи

При использовании криптографических наборов 8 и 9 под симметричным ключом (ключом шифрования и имитозащиты) понимается ключевая информация длины 512 бит, представляющая собой конкатенацию двух независимых криптографических ключей: ключа для шифрования и ключа для имитозащиты. В дальнейшем, с целью сохранения сложившейся в протоколе DLMS терминологии, такая ключевая информация называется ключом.

Типы симметричных ключей, используемых в протоколе DLMS, приведены в таблице 2.

Таблица 2 — Типы симметричных ключей

Тип ключа	Описание	Способ распределения ключа	Применение
Мастер-ключ шифрования и имитозащиты (Key Encryption Key, KEK)	Мастер-ключ, использующийся при передаче (см. 7.2): - (новых) мастер-ключей шифрования и имитозащиты; - глобальных ключей шифрования и имитозащиты при адресной рассылке; - глобальных ключей шифрования и имитозащиты при широковещательной рассылке; - сеансовых ключей шифрования и имитозащиты	По выделенному каналу (описание находится за рамками данного документа) Передача ключа (см. 7.2) Согласование ключа с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа (см. 7.4.1)	Передача ключа посредством: - метода key_transfer; - xDLMS APDU, защищенного способом general-ciphering
Глобальный ключ шифрования и имитозащиты при адресной рассылке (Global unicast encryption key, GUEK)	Ключ шифрования и имитозащиты, использующийся при адресной рассылке для защиты (см. 7.1): - xDLMS APDU; - данных COSEM. Ключ шифрования и имитозащиты при вычислении ответа на запрос в механизме аутентификации «HLS CMAC» (см. 7.5)	Передача ключа (см. 7.2) Согласование ключа с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа (см. 7.4.1)	Защита: - xDLMS APDU способами service-specific global ciphering, general-glo-ciphering, general-ciphering; - данных COSEM с применением объекта Data protection
Глобальный ключ шифрования и имитозащиты при широковещательной рассылке (Global broadcast encryption key, GBEK)	Ключ шифрования и имитозащиты, использующийся при широковещательной рассылке для защиты (см. 7.1): - xDLMS APDU; - данных COSEM	Передача ключа (см. 7.2)	
Специализированный ключ шифрования и имитозащиты (Dedicated key)	Ключ шифрования и имитозащиты, действующий в течение одной установленной АА и использующийся для защиты xDLMS APDU (см. 7.1)	Пересылка ключа в xDLMS InitiateRequest APDU	Защита: - xDLMS APDU способами service-specific dedicated ciphering, general-ded-ciphering

Окончание таблицы 2

Тип ключа	Описание	Способ распределения ключа	Применение
Сеансовый ключ шифрования и имитозащиты (Ephemeral encryption key)	Ключ шифрования и имитозащиты, действующий в течение одного обмена данными и использующийся для защиты (см. 7.1): - xDLMS APDU; - данных COSEM	Передача ключа (см. 7.2) Согласование ключа: - с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа (см. 7.4.2); - с применением каждой из взаимодействующих сторон долговременных ключевых пар согласования ключа (см. 7.4.3)	Защита: - xDLMS APDU способом general-ciphering; - данных COSEM с применением объекта Data protection

5.2 Асимметричные ключи

При использовании криптографического набора 9 под асимметричными ключами понимаются ключи, соответствующие ГОСТ 34.10—2018 и эллиптической кривой, задаваемой набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019.

Типы асимметричных ключей, используемых в протоколе DLMS, приведены в таблице 3.

Т а б л и ц а 3 — Типы асимметричных ключей

Тип ключа	Описание
Ключи подписи и проверки подписи (Digital signature key pair)	Ключи подписи и проверки подписи под - xDLMS APDU (см. 7.3); - данными COSEM (см. 7.3); - сеансовым открытым ключом для согласования ключа (см. 7.4.1, 7.4.2); - сертификатами открытых ключей; - ответом на запрос в механизме аутентификации «HLS GOST34102018-256» (см. 7.5)
Сеансовая ключевая пара для согласования ключа (Ephemeral key agreement key pair)	Сеансовая ключевая пара - у обеих сторон в схеме согласования ключа с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа (см. 7.4.1); - у одной из сторон в схеме согласования ключа с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа (см. 7.4.2)
Долговременная ключевая пара для согласования ключа (Static key agreement key pair)	Долговременная ключевая пара - у обеих сторон в схеме согласования ключа с применением каждой из взаимодействующих сторон долговременных ключевых пар согласования ключа (см. 7.4.3); - у одной из сторон в схеме согласования ключа типа с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа (согласно см. 7.4.2)

Возможны следующие варианты распределения ключа проверки подписи и открытого ключа долговременной ключевой пары для согласования ключа:

- путем внесения сертификатов этих ключей в устройство в процессе его изготовления;
- путем передачи сертификатов этих ключей по выделенному каналу (описание находится за рамками данного документа);
- путем использования методов объекта Security Setup, предназначенных для распределения сертификатов этих ключей (см. 6.4).

Ключ проверки подписи дополнительно может быть распределен в процессе выполнения механизма аутентификации «HLS GOST34102018-256» (см. 7.5) путем пересылки сертификата этого ключа в AARQ или AARE.

Распределение открытого ключа сеансовой ключевой пары осуществляется путем пересылки этого ключа в процессе выполнения алгоритма согласования ключа (см. 7.4.1, 7.4.2).

6 Вспомогательные величины

6.1 Байт управления безопасностью

При шифровании и имитозащите данных в протоколе DLMS используется величина под названием байт управления безопасностью (Security Control byte, SC). Байт управления безопасностью *SC* имеет длину 8 бит и представляется в виде:

$$SC = SC_7||SC_6||\ldots||SC_0 \in V_8,\ SC_i \in V_1,\ I = 0,1,\ldots,7,$$

где строка $SC_3||\ldots||SC_0 = Vec_4(N)$, где N — идентификатор криптографического набора;

бит SC_4 является индикатором применения имитозащиты (0 — не применяется, 1 — применяется);

бит SC_5 является индикатором применения шифрования (0 — данные не шифруются, 1 — данные шифруются);

бит SC_6 является индикатором вида рассылки (0 — адресная, 1 — широковещательная);

бит SC_7 является индикатором применения сжатия данных (в криптографических наборах 8 и 9 сжатие данных не используется и бит SC_7 всегда принимает нулевое значение).

Случай одновременного равенства битов SC_4 и SC_5 нулевому значению подразумевает отсутствие криптографической защиты и в данном документе не рассматривается.

6.2 Идентификатор стороны

Все стороны, использующие протокол DLMS, имеют уникальные имена, называемые идентификаторами сторон (system title). Идентификатор стороны должен иметь длину 64 бита и быть уникальным среди множества всех идентификаторов. Для идентификаторов сторон не требуется обеспечения конфиденциальности.

Некоторые криптографические алгоритмы, описанные в разделе 7, требуют предварительного распределения идентификаторов сторон. Возможны следующие варианты распределения идентификаторов:

- в процессе доверенной регистрации устройства;
- в процессе установления AA между взаимодействующими сторонами;
- путем записи атрибута client_system_title (идентификатор клиента) и чтения атрибута server_system_title (идентификатор сервера) объекта Security setup.

6.3 Счетчик вызовов

Каждой паре: отправитель, ключ шифрования и имитозащиты — ставится в соответствие величина $IC \in V_{32}$, имеющая длину 32 бита и представляющая собой счетчик вызовов операций зашифрования и/или вычисления имитовставки, выполняемых заданным отправителем на заданном ключе.

Перед первой операцией зашифрования и/или вычисления имитовставки на заданном ключе заданный отправитель инициализирует значение *IC* нулем:

$$IC = Vec_{32}(0).$$

При выполнении зашифрования и/или вычисления имитовставки на заданном ключе заданный отправитель использует текущее значение *IC* и затем увеличивает его на единицу, полагая равным величине $Vec_{32}(Int_{32}(IC) + 1)$. В случае, когда исходное значение *IC* равно $Vec_{32}(2^{32} - 1)$, выполнение операции зашифрования и/или вычисления имитовставки считается невозможным, возвращается ошибка и увеличения значения счетчика *IC* не происходит. Последующий порядок обработки данной ситуации определяется конкретным исполнением протокола DLMS. В частности, стороны могут инициировать процедуру смены используемого ключа.

Каждой тройке: отправитель, получатель, ключ шифрования и имитозащиты — ставится в соответствие величина, принимающая значения из $\mathbb{Z}_{2^{32}}$ и представляющая собой минимально допустимое значение счетчика *IC*, которое заданный получатель ожидает получить от заданного отправителя при необходимости расшифрования и/или проверки имитовставки на заданном ключе.

Перед первой операцией расшифрования и/или проверки имитовставки для данных, защищенных заданным отправителем на заданном ключе, заданный получатель инициализирует минимально допустимое значение счетчика *IC* нулем.

При выполнении расшифрования и/или проверки имитовставки для данных, защищенных заданным отправителем на заданном ключе, заданный получатель проверяет, что для входящего значения *IC* справедливо, что $Int_{32}(IC)$ меньше $2^{32} - 1$, но не меньше минимально допустимого значения счетчика. В случае отрицательного результата проверки выполнение операций расшифрования и/или проверки имитовставки считается невозможным и возвращается ошибка. Последующий порядок обработки данной ситуации определяется конкретным исполнением протокола DLMS. В случае положительного результата проверки (и положительного результата проверки имитовставки, в случае ее присутствия) минимально допустимое значение счетчика устанавливается равным $Int_{32}(IC) + 1$.

Для мастер-ключа шифрования и имитозащиты порядок инициализации, изменения и проверки значений соответствующего счетчика аналогичен порядку, описанному выше, с тем отличием, что вместо операций зашифрования и/или вычисления имитовставки (расшифрования и/или проверки имитовставки) рассматривается операция вычисления экспортного представления ключа (вычисления исходного ключа из его экспортного представления).

6.4 Сертификат открытого ключа

При использовании криптографического набора 9 возможно применение сертификатов открытых ключей.

Сертификат открытого ключа должен содержать либо ключ проверки подписи, либо открытый ключ долговременной ключевой пары для согласования ключа. Сертификаты открытых ключей могут применяться для доверенной передачи:

- ключа проверки подписи при защите данных с помощью электронной цифровой подписи (см. 7.3);
- ключей проверки подписи и открытых ключей долговременных ключевых пар для согласования ключа при выполнении согласования ключа (согласно 7.4);
- ключей проверки подписи при выполнении механизма аутентификации «HLS GOST34102018-256» (см. 7.5).

Сертификаты открытых ключей должны иметь формат X.509 версии 3. Ключ, содержащийся в сертификате, должен соответствовать ГОСТ 34.10—2018 и эллиптической кривой, задаваемой набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019. Каждый сертификат должен быть подписан с использованием алгоритма ГОСТ 34.10—2018 на эллиптической кривой, задаваемой набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019. Использование алгоритма ГОСТ 34.10—2018 в сертификате формата X.509 версии 3 должно выполняться в соответствии с Р 1323565.1.023—2018.

Управление сертификатами может осуществляться с применением методов объекта Security Setup, в частности:

- для передачи сертификата по направлению от клиента (или от доверенной третьей стороны) к серверу может использоваться метод import_certificate;
- для передачи сертификата по направлению от сервера к клиенту (или к доверенной третьей стороне) может использоваться метод export_certificate;
- для удаления сертификата, хранящегося на сервере, может использоваться метод remove_certificate;
- для замены сертификата сервера могут использоваться методы generate_key_pair, generate_certificate_request и import_certificate, необходимые, соответственно, для генерации новой ключевой пары сервера, создания сервером запроса на сертификат и передачи созданного сертификата по направлению от клиента (или от доверенной третьей стороны) к серверу.

Описание инфраструктуры открытых ключей, предназначенной для управления сертификатами открытых ключей, находится за рамками данного документа.

7 Описание алгоритмов

В данном разделе описываются криптографические алгоритмы, предназначенные для применения в протоколе DLMS при использовании криптографических наборов 8 и/или 9. Возможные значения аргументов функций в представленных алгоритмах ограничены допустимостью их использования в качестве входных параметров преобразований.

В случае, если описываемые алгоритмы предполагают необходимость генерации случайных значений в процессе своего выполнения, для генерации этих значений должен использоваться датчик случайных чисел, выходные последовательности которого по своим статистическим качествам являются неотличимыми от случайной, равновероятной и независимой последовательности двоичных знаков.

Если для некоторого алгоритма заявляется возможность его применения для защиты xDLMS APDU и данных COSEM, то рассматривается только вариант защиты xDLMS APDU. Защита данных COSEM, то есть атрибутов, параметров метода и возвращаемых значений, осуществляется посредством использования объекта Data protection и применения защиты к APDU, выполняющим роль транспорта при работе с этим объектом.

7.1 Шифрование и имитозащита

Алгоритмы шифрования и имитозащиты предназначены для защиты xDLMS APDU и данных COSEM. Протокол DLMS предполагает наличие алгоритмов шифрования и имитозащиты. Приведенные в данном пункте алгоритмы шифрования и имитозащиты применяются при использовании криптографических наборов 8 и 9.

На вход алгоритму зашифрования и вычисления имитовставки поступают:

- ключ шифрования и имитозащиты $K_{EM} \in V_{512}$,
- ассоциированные данные $AAD \in V^*$,
- открытый текст $Plaintext \in V^*$,
- идентификатор отправителя *originator-system-title* $\in V_{64}$,
- байт управления безопасностью $SC \in V_8$.

Тип ключа шифрования и имитозащиты K_{EM} определяется используемым способом защиты xDLMS APDU:

- general-ciphering,
- general-ded-ciphering,
- general-glo-ciphering,
- service-specific dedicated ciphering,
- service-specific global ciphering.

В случае применения способа защиты general-ciphering ключ K_{EM} должен иметь один из следующих типов:

- глобальный ключ шифрования и имитозащиты при адресной рассылке;
- глобальный ключ шифрования и имитозащиты при широковещательной рассылке;
- сеансовый ключ шифрования и имитозащиты.

В случае применения способов защиты general-glo-ciphering и service-specific global ciphering ключ K_{EM} должен иметь один из следующих типов:

- глобальный ключ шифрования и имитозащиты при адресной рассылке;
- глобальный ключ шифрования и имитозащиты при широковещательной рассылке.

В случае применения способов защиты general-ded-ciphering и service-specific dedicated ciphering ключ K_{EM} должен являться специализированным ключом шифрования и имитозащиты.

Структура ассоциированных данных *AAD* и открытого текста *Plaintext* определяется значением байта управления безопасности *SC*:

- в случае $SC_5 = 0$, $SC_4 = 1$:

$$AAD = SC \,||\, AF \,||\, Unprotected\_APDU,$$

$$Plaintext \in V_0,$$

- в случае $SC_5 = 1$, $SC_4 = 0$:

$$AAD \in V_0,$$

$$Plaintext = Unprotected\_APDU,$$

- в случае $SC_5 = 1$, $SC_4 = 0$:

$$AAD = SC \,||\, AF,$$

$$Plaintext = Unprotected\_APDU,$$

где $Unprotected\_APDU \in V^*$ — защищаемый xDLMS APDU, $AF \in V^*$ — дополнительно защищаемые данные, определяемые используемым способом защиты xDLMS APDU.

В случае применения способа защиты general-ciphering значение *AF* должно иметь вид

AF = transaction-id||originator-system-title||recipient-system-title||date-time||other-information,

где *transaction-id* — значение параметра transaction-id (идентификатор транзакции) xDLMS APDU, защищенного способом general-ciphering;

originator-system-title — значение параметра originator-system-title (идентификатор отправителя) xDLMS APDU, защищенного способом general-ciphering;

recipient-system-title — значение параметра recipient-system-title (идентификатор получателя) xDLMS APDU, защищенного способом general-ciphering;

date-time — значение параметра date-time (дата и время отправления) xDLMS APDU, защищенного способом general-ciphering;

other-information — значение параметра other-information (прочая информация) xDLMS APDU, защищенного способом general-ciphering.

В случае применения способов защиты general-ded-ciphering, general-glo-ciphering, service-specific dedicated ciphering, service-specific global ciphering значение $AF \in V_0$.

Для осуществления зашифрования и вычисления имитовставки формируется значение синхропосылки $IV_{EM} \in V_{96}$:

$$IV_{EM} = originator\text{-}system\text{-}title \,||\, IC_{EM},$$

где $IC_{EM} \in V_{32}$ — значение счетчика вызовов, уникальное для каждой операции зашифрования и/или вычисления имитовставки, выполняемой стороной с идентификатором *originator-system-title* с использованием ключа K_{EM}. Порядок инициализации, изменения и проверки значений счетчика вызовов описан в 6.3.

Дальнейшие вычисления определяются значениями бит SC_5, SC_4:

- при $SC_5 = 0$, $SC_4 = 1$ вычисляется

$$AuthTag = KUZN\_CMAC(K_M, IV_{EM} \,||\, AAD) \in V_{96},$$

где $K_M = LSB_{256}(K_{EM}) \in V_{256}$;

- при $SC_5 = 1$, $SC_4 = 0$ вычисляется

$$Ciphertext = KUZN\_CTR(K_E, IV_{EM}, Plaintext) \in V_{|Plaintext|},$$

где $K_E = MSB_{256}(K_{EM}) \in V_{256}$;

- при $SC_5 = 1$, $SC_4 = 1$ вычисляются

$$Ciphertext = KUZN\_CTR(K_E, IV_{EM}, Plaintext) \in V_{|Plaintext|},$$

$$AuthTag = KUZN\_CMAC(K_M, IV_{EM} \,||\, AAD \,||\, Ciphertext) \in V_{96},$$

где $K_E = MSB_{256}(K_{EM}) \in V_{256}$, $K_M = LSB_{256}(K_{EM}) \in V_{256}$.

Выходными значениями алгоритмов зашифрования и вычисления имитовставки являются *Ciphertext* и/или *AuthTag*, полученные на предыдущем шаге.

7.2 Передача ключа

Алгоритм передачи ключа предназначен для передачи ключа шифрования и имитозащиты по направлению от одной стороны к другой. Приведенный в данном пункте алгоритм передачи ключа применяется при использовании криптографических наборов 8 и 9.

Пошаговое описание алгоритма передачи ключа шифрования и имитозащиты $Key \in V_{512}$ от стороны *U* к стороне *V* приведено в таблице 4.

Для возможности выполнения алгоритма передачи ключа:

- сторона *U* и сторона *V* должны иметь общий мастер-ключ шифрования и имитозащиты $K_{KEK} \in V_{512}$;
- сторона *V* должна предварительно получить значение идентификатора стороны *U*: $system\text{-}title\text{-}U \in V_{64}$.

Передаваемый ключ *Key* должен иметь один из следующих типов:

- мастер-ключ шифрования и имитозащиты,
- глобальный ключ шифрования и имитозащиты при адресной рассылке,
- глобальный ключ шифрования и имитозащиты при широковещательной рассылке,
- сеансовый ключ шифрования и имитозащиты.

При вычислении синхропосылки $IV_{KEK} \in V_{96}$ сторона *U* использует значение счетчика вызовов $IC_{KEK} \in V_{32}$, уникальное для каждой операции передачи ключа, выполняемой стороной *U* с использова-

нием ключа K_{KEK}. Порядок инициализации, изменения и проверки значений счетчика IC_{KEK} аналогичен порядку, описанному в 6.3, с тем отличием, что вместо операций зашифрования и/или вычисления имитовставки (расшифрования и/или проверки имитовставки) рассматривается операция вычисления экспортного представления ключа (вычисления исходного ключа из его экспортного представления).

Способы обработки сбоев и ошибок в процессе передачи ключа определяются конкретным исполнением протокола DLMS.

Таблица 4 — Передача ключа от стороны *U* к стороне *V*

Сторона *U*	Пересылаемые значения	Сторона *V*
Вычисляет синхропосылку длины 96 бит: $IV_{KEK} = system\text{-}title\text{-}U \,||\, IC_{KEK} \in V_{96}$		
Вычисляет мастер-ключ шифрования длины 256 бит: $K_{KEKE} = MSB_{256}(K_{KEK}) \in V_{256}$		
Вычисляет мастер-ключ имитозащиты длины 256 бит: $K_{KEKM} = LSB_{256}(K_{KEK}) \in V_{256}$		
Вычисляет экспортное представление передаваемого ключа, имеющее длину 640 бит: $ExpKey = KUZN\_KEXP (Key, K_{KEKM}, K_{KEKE}, IV_{KEK}) \in V_{640}$		
	IC_{KEK}, $ExpKey$ →	
		Проверяет, что для значения счетчика IC_{KEK} справедливо, что $Int_{32}(IC_{KEK})$ меньше $2^{32}–1$, но не меньше минимально допустимого значения счетчика. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс передачи ключа
		Вычисляет синхропосылку длины 96 бит: $IV_{KEK} = system\text{-}title\text{-}U \,||\, IC_{KEK} \in V_{96}$
		Вычисляет мастер-ключ шифрования длины 256 бит: $K_{KEKE} = MSB_{256}(K_{KEK}) \in V_{256}$
		Вычисляет мастер-ключ имитозащиты длины 256 бит: $K_{KEKM} = LSB_{256}(K_{KEK}) \in V_{256}$
		Вычисляет передаваемый ключ длины 512 бит: $Key = KUZN\_KIMP (ExpKey, K_{KEKM}, K_{KEKE}, IV_{KEK}) \in V_{512}$ (в случае отрицательного результата проверки имитовставки возвращает ошибку и досрочно завершает процесс передачи ключа)

При передаче мастер-ключа шифрования и имитозащиты, глобального ключа шифрования и имитозащиты при адресной рассылке или глобального ключа шифрования и имитозащиты при широковещательной рассылке в качестве стороны *U* выступает клиент, а в качестве стороны *V* — сервер. В этом случае для пересылки значений IC_{KEK}, *ExpKey* используется метод key_transfer, для чего устанавливаются следующие значения его параметров:

- тип передаваемого ключа key_id ∈ {0,1,3},

где 0 — глобальный ключ шифрования и имитозащиты при адресной рассылке, 1 — глобальный ключ шифрования и имитозащиты при широковещательной рассылке, 3 — мастер-ключ шифрования и имитозащиты;

- экспортное представление ключа и сопутствующая информация:

$$key\_wrapped = IC_{KEK} \,||\, ExpKey.$$

При передаче сеансового ключа шифрования и имитозащиты в качестве одной из взаимодействующих сторон *U* или *V* выступает клиент, в качестве другой стороны — сервер. В этом случае для пересылки значений IC_{KEK}, *ExpKey* используется xDLMS APDU, защищенный способом general-ciphering, для чего устанавливаются следующие значения его параметров:

- тип мастер-ключа:

key-info.wrapped-key.kek-id = 0,

где 0 — мастер-ключ шифрования и имитозащиты;

- экспортное представление ключа и сопутствующая информация:

key-info.wrapped-key.key-ciphered-data = $IC_{KEK} \,||\, ExpKey$.

Пересылаемый xDLMS APDU шифруется и/или имитозащищается с использованием передаваемого ключа *Key*.

Следует отметить, что процесс передачи специализированного ключа шифрования и имитозащиты отличается от описанного в таблице 4. Данный ключ передается без вычисления его экспортного представления путем пересылки в xDLMS InitiateRequest APDU, который в обязательном порядке шифруется и имитозащищается на глобальном ключе шифрования и имитозащиты при адресной рассылке.

7.3 Защита с помощью электронной цифровой подписи

Алгоритм защиты с помощью электронной цифровой подписи предназначен для защиты xDLMS APDU и данных COSEM. Приведенный в данном пункте алгоритм защиты с помощью электронной цифровой подписи применяется при использовании криптографического набора 9.

На вход алгоритму вычисления электронной цифровой подписи поступают:

- ключ подписи $d_{sign} \in \{1,2,\ldots,q\text{-}1\}$,
- дополнительно защищаемые данные $AF \in V^*$,
- защищаемые данные $Content \in V^*$.

Дополнительно защищаемые данные *AF* и защищаемые данные *Content* определяются способом защиты xDLMS APDU general-signing.

Значение *Content* должно иметь вид

$$Content = Unprotected\_APDU,$$

где $Unprotected\_APDU \in V^*$ — защищаемый xDLMS APDU.

Значение *AF* должно иметь вид

$$AF = transaction\text{-}id||originator\text{-}system\text{-}title||recipient\text{-}system\text{-}title||date\text{-}time||other\text{-}information,$$

где *transaction-id* — значение параметра transaction-id (идентификатор транзакции) xDLMS APDU, защищенного способом general-signing;

originator-system-title — значение параметра originator-system-title (идентификатор отправителя) xDLMS APDU, защищенного способом general-signing;

recipient-system-title — значение параметра recipient-system-title (идентификатор получателя) xDLMS APDU, защищенного способом general-signing;

date-time — значение параметра date-time (дата и время отправления) xDLMS APDU, защищенного способом general-signing;

other-information — значение параметра other-information (прочая информация) xDLMS APDU, защищенного способом general-signing.

Выходным значением алгоритма вычисления электронной цифровой подписи является подпись *Signature* длины 512 бит, вычисляемая следующим образом:

$$Signature = SIGN_{256}(d_{sign}, AF \,||\, Content) \in V_{512}.$$

Ключи подписи и проверки подписи, используемые при защите данных с помощью электронной цифровой подписи, должны удовлетворять ГОСТ 34.10—2018 и эллиптической кривой, задаваемой набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019.

7.4 Согласование ключа

Алгоритмы согласования ключа предназначены для согласования ключа шифрования и имитозащиты между двумя сторонами. Приведенные в данном пункте алгоритмы согласования ключа применяются при использовании криптографического набора 9.

Рассматриваются три алгоритма согласования ключа:

- с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа (см. 7.4.1);
- с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа (см. 7.4.2);
- с применением каждой из взаимодействующих сторон долговременных ключевых пар согласования ключа (см. 7.4.3).

В таблице 5 приведены основные свойства каждого из перечисленных алгоритмов согласования ключа, а также алгоритма передачи ключа, описанного в 7.2.

Т а б л и ц а 5 — Алгоритмы распределения ключа шифрования и имитозащиты

Свойства алгоритма	Согласование ключа на сеансовых ключах (см. 7.4.1)	Согласование ключа на сеансовом и долговременном ключе (см. 7.4.2)	Согласование ключа на долговременных ключах (см. 7.4.3)	Передача ключа (см. 7.2)
Криптографические наборы, поддерживающие алгоритм	9	9	9	8, 9
Возможные типы распределяемого ключа	- мастер-ключ шифрования и имитозащиты; - глобальный ключ шифрования и имитозащиты при адресной рассылке	- сеансовый ключ шифрования и имитозащиты	- сеансовый ключ шифрования и имитозащиты	- мастер-ключ шифрования и имитозащиты; - глобальный ключ шифрования и имитозащиты при адресной рассылке; - глобальный ключ шифрования и имитозащиты при широковещательной рассылке; - сеансовый ключ шифрования и имитозащиты
Используемые в процессе вычислений ключи	У каждой из сторон должны быть: - ключи подписи и проверки подписи; - сеансовая ключевая пара согласования ключа	У одной стороны должны быть: - ключи подписи и проверки подписи; - сеансовая ключевая пара согласования ключа У другой стороны должна быть: - долговременная ключевая пара согласования ключа	У каждой из сторон должна быть: - долговременная ключевая пара согласования ключа	У каждой из сторон должен быть: - мастер-ключ шифрования и имитозащиты
Тип взаимодействия (первым указывается инициатор)	- клиент-сервер	- клиент-сервер; - сервер-клиент; - доверенная третья сторона-сервер; - сервер-доверенная третья сторона	- клиент-сервер; - сервер-клиент; - доверенная третья сторона-сервер; - сервер-доверенная третья сторона	- клиент-сервер; - сервер-клиент (только при передаче сеансового ключа шифрования и имитозащиты)

Используемые в процессе согласования ключа сеансовые и долговременные ключевые пары для согласования ключа, ключи подписи и проверки подписи должны соответствовать ГОСТ 34.10—2018

и эллиптической кривой, задаваемой набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019.

Для возможности выполнения алгоритма согласования ключа между сторонами *U* и *V*:

- сторона *V* должна предварительно получить значение идентификатора стороны *U*: *system-title-U* $\in V_{64}$;
- сторона *U* должна предварительно получить значение идентификатора стороны *V*: *system-title-V* $\in V_{64}$.

В процессе согласования ключа стороны используют величину *AlgorithmID*, определяющую криптографические алгоритмы, для которых предназначен согласуемый ключ. Возможные значения *AlgorithmID* приведены в таблице 6. При подаче величины *AlgorithmID* на вход криптографическим алгоритмам необходимо использовать ее закодированное значение длиной 56 бит.

Способы обработки сбоев и ошибок в процессе согласования ключа определяются конкретным исполнением протокола DLMS.

Таблица 6 — Значения *AlgorithmID*

Криптографические алгоритмы, для которых предназначен согласуемый ключ	Идентификатор криптографического алгоритма COSEM (COSEM cryptographic algorithm ID)	Закодированное значение (в виде шестнадцатеричной строки)
Алгоритмы шифрования и имитозащиты *KUZN_CTR* и *KUZN_CMAC* (выбирается в случае согласования глобального ключа шифрования и имитозащиты при адресной рассылке или сеансового ключа шифрования и имитозащиты)	2.16.756.5.8.3.4	60857406080304
Алгоритмы экспорта и импорта ключа *KUZN_KEXP* и *KUZN_KIMP* (выбирается в случае согласования мастер-ключа шифрования и имитозащиты)	2.16.756.5.8.3.5	60857406080305

7.4.1 Согласование ключа с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа

Алгоритм согласования ключа между сторонами *U* и *V* с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа приведен в таблице 8. В качестве стороны *U* выступает клиент, в качестве стороны *V* — сервер.

Для возможности выполнения алгоритма согласования ключа с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа:

- сторона *U* должна иметь собственные долговременные ключи подписи и проверки подписи: $d_{s,U}^{sign}$ и $Q_{s,U}^{sign}$ соответственно, где $Q_{s,U}^{sign} = d_{s,U}^{sign}P$;
- сторона *V* должна иметь собственные долговременные ключи подписи и проверки подписи: $d_{s,V}^{sign}$ и $Q_{s,V}^{sign}$ соответственно, где $Q_{s,V}^{sign} = d_{s,V}^{sign}P$;
- стороны должны предварительно обменяться ключами $Q_{s,U}^{sign}$, $Q_{s,V}^{sign}$ доверенным образом (например, с помощью использования сертификатов).

Согласуемый ключ *Key* $\in V_{512}$ должен иметь один из следующих типов:

- мастер-ключ шифрования и имитозащиты,
- глобальный ключ шифрования и имитозащиты при адресной рассылке.

В процессе согласования ключа стороны используют величину *key_id*, определяющую тип согласуемого ключа. Возможные значения *key_id* приведены в таблице 7. Конкретное значение *key_id* определяется стороной *U* перед выполнением алгоритма согласования ключа. При подаче величины *key_id* на вход криптографическим алгоритмам необходимо использовать ее закодированное значение длиной 16 бит.

Таблица 7 — Значения *key_id*

Тип согласуемого ключа	Значение	Закодированное значение (в виде шестнадцатеричной строки)
Глобальный ключ шифрования и имитозащиты при адресной рассылке	0	1600
Мастер-ключ шифрования и имитозащиты	3	1603

Таблица 8 — Согласование ключа с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа

Сторона *U*	Пересылаемые значения	Сторона *V*
Генерирует сеансовую ключевую пару согласования ключа ($d_{e,U}^{agr}$, $Q_{e,U}^{agr}$), где $d_{e,U}^{agr}$ — закрытый ключ, $Q_{e,U}^{agr}$ — открытый ключ, для чего: случайным образом выбирает значение $d_{e,U}^{agr} \in \{1, 2,\ldots,q-1\}$ и вычисляет точку эллиптической кривой $Q_{e,U}^{agr} = d_{e,U}^{agr}P$		
	key_id, $Q_{e,U}^{agr}$ →	
		Проверяет, что $Q_{e,U}^{agr} \in E$ и $\frac{m}{q}Q_{e,U}^{agr} \neq \mathcal{O}$. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа
		Генерирует сеансовую ключевую пару согласования ключа ($d_{e,V}^{agr}$, $Q_{e,V}^{agr}$), где $d_{e,V}^{agr}$ — закрытый ключ, $Q_{e,V}^{agr}$ — открытый ключ, для чего: случайным образом выбирает значение $d_{e,V}^{agr} \in \{1, 2,\ldots,q-1\}$ и вычисляет точку эллиптической кривой $Q_{e,V}^{agr} = d_{e,V}^{agr}P$
	← $Q_{e,V}^{agr}$	
Проверяет, что $Q_{e,V}^{agr} \in E$ и $\frac{m}{q}Q_{e,V}^{\arg} \neq \mathcal{O}$. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа		
Вычисляет значение P_{VU} длины 256 бит: $P_{VU} = VKO_{256}(d_{e,U}^{agr}, Q_{e,V}^{agr}, 1) \in V_{256}$		
Вычисляет значение T_{VU} длины 768 бит: $T_{VU} = KDFTREE(P_{VU}, AlgorithmID, system\text{-}title\text{-}U \,||\, system\text{-}title\text{-}V, 1) \in V_{768}$		
Вычисляет значение K_{VU} длины 512 бит и значение M_{VU} длины 256 бит: $K_{VU} = LSB_{512}(T_{VU}) \in V_{512}$, $M_{VU} = MSB_{256}(T_{VU}) \in V_{256}$		
Вычисляет цифровую подпись $sign_U$ длины 512 бит: $sign_U = SIGN_{256}(d_{s,U}^{sign}, key\_id \,||\, \pi_x(Q_{e,U}^{agr}) \,||\, \pi_x(Q_{e,V}^{agr}) \,||\, system\text{-}title\text{-}V) \in V_{512}$		

Окончание таблицы 8

Сторона *U*	Пересылаемые значения	Сторона *V*
Вычисляет значение tag_U длины 96 бит: $tag_U = KUZN\_CMAC(M_{VU}, \pi_x(Q_{e,U}^{agr})\|\|\pi_x(Q_{e,V}^{agr})\|\| system\text{-}title\text{-}U\|\|system\text{-}title\text{-}V) \in V_{96}$		
	$sign_U$, tag_U →	
		Проверяет цифровую подпись $sign_U$ с использованием $Q_{s,U}^{sign}$. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа
		Вычисляет значение P_{UV} длины 256 бит: $P_{UV} = VKO_{256}(d_{e,V}^{agr}, Q_{e,U}^{agr}, 1) \in V_{256}$
		Вычисляет значение T_{UV} длины 768 бит: $T_{UV} = KDFTREE(P_{UV}, AlgorithmID, system\text{-}title\text{-}U \| system\text{-}title\text{-}V, 1) \in V_{768}$
		Вычисляет значение K_{UV} длины 512 бит и значение M_{UV} длины 256 бит: $K_{UV} = LSB_{512}(T_{UV}) \in V_{512}$, $M_{UV} = MSB_{256}(T_{UV}) \in V_{256}$
		Проверяет равенство $tag_U = KUZN\_CMAC(M_{UV}, \pi_x(Q_{e,U}^{agr})\|\|\pi_x(Q_{e,V}^{agr})$ ||system-title-U||system-title-V). В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа
		Вычисляет цифровую подпись $sign_V$ длины 512 бит: $sign_V = SIGN_{256}(d_{s,V}^{sign}, \pi_x(Q_{e,V}^{agr})\|\| \pi_x(Q_{e,U}^{agr})\|\|system\text{-}title\text{-}U) \in V_{512}$
		Вычисляет значение tag_V длины 96 бит: $tag_V = KUZN\_CMAC(M_{UV}, \pi_x(Q_{e,V}^{agr})\|\| \pi_x(Q_{e,U}^{agr})\|\|system\text{-}title\text{-}V\|\|system\text{-}title\text{-}U) \in V_{96}$
		Уничтожает ключ M_{UV} и устанавливает значение *Key* длины 512 бит: $Key = K_{UV} \in V_{512}$
	← $sign_V$, tag_V	
Проверяет цифровую подпись $sign_V$ с использованием $Q_{s,V}^{sign}$. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа		
Проверяет равенство $tag_V = KUZN\_CMAC(M_{VU}, \pi_x(Q_{e,V}^{agr})\|\| \pi_x(Q_{e,U}^{agr})\|\|system\text{-}title\text{-}V\|\|system\text{-}title\text{-}U)$. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа		
Уничтожает ключ M_{VU} и устанавливает значение *Key* длины 512 бит: $Key = K_{VU} \in V_{512}$		

Для пересылки значений *key_id*, $Q_{e,U}^{agr}$ используется метод key_agreement, для чего устанавливаются следующие значения его параметров:

- тип согласуемого ключа:

$$\text{key\_id} = key\_id \in \{0,3\};$$

- сеансовый открытый ключ и сопутствующая информация:

$$\text{key\_data} = \pi_x(Q_{e,U}^{agr}) \,||\, \pi_y(Q_{e,U}^{agr}).$$

Для пересылки значения $Q_{e,V}^{agr}$ используется метод key_agreement, для чего устанавливаются следующие значения его параметров:

- тип согласуемого ключа:

$$\text{key\_id} = key\_id \in \{0,3\};$$

- сеансовый открытый ключ и сопутствующая информация:

$$\text{key\_data} = \pi_x(Q_{e,V}^{agr}) \,||\, \pi_y(Q_{e,V}^{agr}).$$

Для пересылки значений $sign_U$, tag_U используется метод key_agreement_confirmation, описанный в 7.4.1.1, для чего устанавливаются следующие значения его параметров:

тип согласуемого ключа:

$$\text{key\_id} = key\_id \in \{0,3\};$$

информация для подтверждения согласуемого ключа и взаимодействующих сторон:

$$\text{key\_confirmation\_data} = sign_U \,||\, tag_U.$$

Для пересылки значений $sign_V$, tag_V используется метод key_agreement_confirmation, описанный в 7.4.1.1, для чего устанавливаются следующие значения его параметров:

- тип согласуемого ключа:

$$\text{key\_id} = key\_id \in \{0,3\};$$

- информация для подтверждения согласуемого ключа и взаимодействующих сторон:

$$\text{key\_confirmation\_data} = sign_V \,||\, tag_V.$$

7.4.1.1 Описание метода подтверждения ключа и взаимодействующих сторон

Для возможности подтверждения согласуемого ключа и взаимодействующих сторон вводится дополнительный метод key_agreement_confirmation объекта Security Setup. Описание метода в соответствии с обозначениями, принятыми в [2], приведено ниже.

```
key_agreement_confirmation (data)

data ::= array key_agreement_confirmation_data

key_agreement_confirmation_data ::= structure
{
    key_id: enum:
      (0) global unicast encryption key,
      (3) master key (KEK)
    key_confirmation_data: octet-string
}
```

Метод key_agreement_confirmation имеет один входной параметр data. Значение data представляет собой массив, состоящий из структур, каждая из которых содержит два элемента:

- тип согласуемого ключа key_id, который может принимать значение 0 в случае согласования глобального ключа шифрования и имитозащиты при адресной рассылке, и значение 3 в случае согласования мастер-ключа шифрования и имитозащиты;

- информацию для подтверждения согласуемого ключа и взаимодействующих сторон key_confirmation_data, которая должна содержать значения подписи и имитовставки, вычисленные с использованием сеансовых ключевых пар согласования ключа, идентификаторов сторон и долговременного ключа подписи.

7.4.2 Согласование ключа с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа

Алгоритм согласования ключа между сторонами *U* и *V* с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа приведен в таблице 9. В качестве одной из взаимодействующих сторон *U* или *V* выступает клиент (или доверенная третья сторона), в качестве другой стороны — сервер.

Для возможности выполнения алгоритма согласования ключа с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа:

- сторона *U* должна иметь собственные долговременные ключи подписи и проверки подписи: $d_{s,U}^{sign}$ и $Q_{s,U}^{sign}$ соответственно, где $Q_{s,U}^{sign} = d_{s,U}^{sign}P$;
- сторона *V* должна иметь собственную долговременную ключевую пару согласования ключа ($d_{s,V}^{agr}$, $Q_{s,V}^{agr}$), где $d_{s,V}^{agr}$ — закрытый ключ, $Q_{e,V}^{agr}$ — открытый ключ и $Q_{s,V}^{agr} = d_{s,V}^{agr}P$;
- сторона *V* должна предварительно получить ключ $Q_{s,U}^{sign}$ доверенным образом (например, с помощью использования сертификата);
- сторона *U* должна предварительно получить ключ $Q_{s,V}^{agr}$ доверенным образом (например, с помощью использования сертификата).

Согласуемый ключ $Key \in V_{512}$ должен являться сеансовым ключом шифрования и имитозащиты.

Таблица 9 — Согласование ключа с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа

Сторона *U*	Пересылаемые значения	Сторона *V*
Генерирует случайное значение $r_U \in V_{128}$. В случае $r_U = 0^{128}$ устанавливает $r_U = 0^{127} \| 1$		
Генерирует сеансовую ключевую пару согласования ключа ($d_{e,U}^{agr}$, $Q_{e,U}^{agr}$), где $d_{e,U}^{agr}$ — закрытый ключ, $Q_{e,U}^{agr}$ — открытый ключ, для чего: случайным образом выбирает значение $d_{e,U}^{agr} \in \{1, 2,\ldots,q-1\}$ и вычисляет точку эллиптической кривой $Q_{e,U}^{agr} = d_{e,U}^{agr}P$		
Вычисляет значение P_{VU} длины 256 бит: $P_{VU} = VKO_{256}(d_{e,U}^{agr}, Q_{s,V}^{agr}, r_U) \in V_{256}$		
Вычисляет значение T_{VU} длины 768 бит: $T_{VU} = KDFTREE(P_{VU}, AlgorithmID,$ *system-title-U* $\|\|$ *system-title-V,1)* $\in V_{768}$		
Вычисляет значение K_{VU} длины 512 бит и значение M_{VU} длины 256 бит: $K_{VU} = LSB_{512}(T_{VU}) \in V_{512}$, $M_{VU} = MSB_{256}(T_{VU}) \in V_{256}$		
Вычисляет цифровую подпись $sign_U$ длины 512 бит: $sign_U = SIGN_{256}(d_{s,U}^{sign}, \pi_x(Q_{e,U}^{agr}) \| \pi_x(Q_{s,V}^{agr}) \|$ *system-title-V)* $\in V_{512}$		
Вычисляет значение tag_U длины 96 бит: $tag_U = KUZN\_CMAC(M_{VU}, r_U \| \pi_x(Q_{e,U}^{agr} \| \pi_x(Q_{s,V}^{agr}) \|$ *system-title-U* $\|\|$ *system-title-V)* $\in V_{96}$		

Окончание таблицы 9

Сторона *U*	Пересылаемые значения	Сторона *V*
Уничтожает ключ M_{VU} и устанавливает значение *Key* длины 512 бит: $Key = K_{VU} \in V_{512}$		
	r_U, $Q_{e,U}^{agr}$, $sign_U$, tag_U →	
		Проверяет, что $Q_{e,U}^{agr} \in E$ и $\frac{m}{q} Q_{e,U}^{arg} \neq \mathcal{O}$. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа
		Проверяет цифровую подпись $sign_U$ с использованием $Q_{s,U}^{sign}$. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа
		Вычисляет значение P_{UV} длины 256 бит: $P_{UV} = VKO_{256}(d_{s,V}^{agr}, Q_{e,U}^{agr}, r_U) \in V_{256}$
		Вычисляет значение T_{UV} длины 768 бит: $T_{UV} = KDFTREE(P_{UV}, AlgorithmID, system\text{-}title\text{-}U \| system\text{-}title\text{-}V, 1) \in V_{768}$
		Вычисляет значение K_{UV} длины 512 бит и значение M_{UV} длины 256 бит: $K_{UV} = LSB_{512}(T_{UV}) \in V_{512}$, $M_{UV} = MSB_{256}(T_{UV}) \in V_{256}$
		Проверяет равенство $tag_U = KUZN\_CMAC(M_{UV}, r_U \| \pi_x(Q_{e,U}^{agr}) \| \pi_x(Q_{s,V}^{agr}) \| system\text{-}title\text{-}U \| system\text{-}title\text{-}V)$. В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа
		Уничтожает ключ M_{UV} и устанавливает значение *Key* длины 512 бит: $Key = K_{UV} \in V_{512}$

Для пересылки значений r_U, $Q_{e,U}^{agr}$, $sign_U$, tag_U используется xDLMS APDU, защищенный способом general-ciphering, для чего устанавливаются следующие значения его параметров:

- идентификатор транзакции:

transaction-id = r_U;

- идентификатор схемы согласования ключа:

key-info.agreed-key.key-parameters = 01 (в шестнадцатеричном виде);

- дополнительная информация для согласования ключа:

key-info.agreed-key.key-ciphered-data = $\pi_x(Q_{e,U}^{agr}) \| \pi_y(Q_{e,U}^{agr}) \| sign_U \| tag_U$.

Одновременно с передачей перечисленных значений, необходимых для возможности вычисления ключа *Key* стороной *V*, в этом же xDLMS APDU передается содержательная часть сообщения, зашифрованная и/или имитозащищенная с использованием согласуемого ключа *Key*.

В случае необходимости обеспечения защиты от повторов при передаче xDLMS APDU, защищенного на согласуемом сеансовом симметричном ключе, взаимодействующие стороны должны предусмотреть использование дополнительного механизма защиты. Такой механизм, к примеру, может базироваться на использовании меток времени, при котором:

- сторона *U* устанавливает значение поля date-time (дата и время) xDLMS APDU, защищенного способом general-ciphering, равным значению времени отправки этого xDLMS APDU;

- сторона *U* в обязательном порядке выполняет имитозащиту xDLMS APDU;

- сторона *V* при получении xDLMS APDU проверяет имитовставку для xDLMS APDU и сверяет значение поля date-time с текущим значением времени; на основании этих проверок сторона *V* принимает решение о необходимости приема полученного xDLMS APDU.

Описание конкретного механизма защиты от повторов при передаче xDLMS APDU, защищенного на согласуемом сеансовом симметричном ключе, находится за рамками данного документа.

7.4.3 Согласование ключа с применением каждой из взаимодействующих сторон долговременных ключевых пар согласования ключа

Алгоритм согласования ключа между сторонами *U* и *V* с применением каждой из взаимодействующих сторон долговременных ключевых пар согласования ключа приведен в таблице 10. В качестве одной из взаимодействующих сторон *U* или *V* выступает клиент (или доверенная третья сторона), в качестве другой стороны — сервер.

Для возможности выполнения алгоритма согласования ключа с применением каждой из взаимодействующих сторон долговременных ключевых пар согласования ключа:

- сторона *U* должна иметь собственную долговременную ключевую пару согласования ключа ($d_{s,U}^{agr}$, $Q_{s,U}^{agr}$), где $d_{s,U}^{agr}$ — закрытый ключ, $Q_{s,U}^{agr}$ — открытый ключ и $Q_{s,U}^{agr} = d_{s,U}^{agr}P$;

- сторона *V* должна иметь собственную долговременную ключевую пару согласования ключа ($d_{s,V}^{agr}$, $Q_{s,V}^{agr}$), где $d_{s,V}^{agr}$ — закрытый ключ, $Q_{s,V}^{agr}$ — открытый ключ и $Q_{s,V}^{agr} = d_{s,V}^{agr}P$;

- стороны *U* и *V* должны предварительно обменяться ключами $Q_{s,U}^{agr}$, $Q_{s,V}^{agr}$ доверенным образом (например, с помощью использования сертификатов).

Согласуемый ключ $Key \in V_{512}$ должен являться сеансовым ключом шифрования и имитозащиты.

Таблица 10 — Согласование ключа с применением каждой из взаимодействующих сторон долговременных ключевых пар согласования ключа

Сторона *U*	Пересылаемые значения	Сторона *V*
Генерирует случайное значение $r_U \in V_{128}$. В случае $r_U = 0^{128}$ устанавливает $r_U = 0^{127} \| 1$		
Вычисляет значение P_{VU} длины 256 бит: $P_{VU} = VKO_{256}(d_{s,U}^{agr}, Q_{s,V}^{agr}, r_U) \in V_{256}$		
Вычисляет значение T_{VU} длины 768 бит: $T_{VU} = KDFTREE(P_{VU}, AlgorithmID, system\text{-}title\text{-}U \| system\text{-}title\text{-}V, 1) \in V_{768}$		
Вычисляет значение K_{VU} длины 512 бит и значение M_{VU} длины 256 бит: $K_{VU} = LSB_{512}(T_{VU}) \in V_{512}$, $M_{VU} = MSB_{256}(T_{VU}) \in V_{256}$		
Вычисляет значение tag_U длины 96 бит: $tag_U = KUZN\_CMAC(M_{VU}, r_U \| \pi_x(Q_{s,U}^{agr}) \| \pi_x(Q_{s,V}^{agr}) \| system\text{-}title\text{-}U \| system\text{-}title\text{-}V) \in V_{96}$		
Уничтожает ключ M_{VU} и устанавливает значение *Key* длины 512 бит: $Key = K_{VU} \in V_{512}$		
	r_U, tag_U →	
		Вычисляет значение P_{UV} длины 256 бит: $P_{UV} = VKO_{256}(d_{s,V}^{agr}, Q_{s,U}^{agr}, r_U) \in V_{256}$
		Вычисляет значение T_{UV} длины 768 бит: $T_{UV} = KDFTREE(P_{UV}, AlgorithmID, system\text{-}title\text{-}U \| system\text{-}title\text{-}V, 1) \in V_{768}$

Окончание таблицы 10

Сторона *U*	Пересылаемые значения	Сторона *V*
		Вычисляет значение K_{UV} длины 512 бит и значение M_{UV} длины 256 бит: $K_{UV} = LSB_{512}(T_{UV}) \in V_{512}$, $M_{UV} = MSB_{256}(T_{UV}) \in V_{256}$
		Проверяет равенство $tag_U = KUZN\_CMAC(M_{UV}, r_U \| \pi_x(Q^{agr}_{s,U}) \| \pi_x(Q^{agr}_{s,V}) \|$ *system-title-U* $\|$ *system-title-V*). В случае отрицательного результата проверки возвращает ошибку и досрочно завершает процесс согласования ключа
		Уничтожает ключ M_{UV} и устанавливает значение *Key* длины 512 бит: $Key = K_{UV} \in V_{512}$

Для пересылки значений r_U, tag_U используется xDLMS APDU, защищенный способом general-ciphering, для чего устанавливаются следующие значения его параметров:

- идентификатор транзакции:

$$\text{transaction-id} = r_U;$$

- идентификатор схемы согласования ключа:

key-info.agreed-key.key-parameters = 02 (в шестнадцатеричном виде);

- дополнительная информация для согласования ключа:

$$\text{key-info.agreed-key.key-ciphered-data} = tag_U.$$

Одновременно с передачей перечисленных значений, необходимых для возможности вычисления ключа *Key* стороной *V*, в этом же xDLMS APDU передается содержательная часть сообщения, зашифрованная и/или имитозащищенная с использованием согласуемого ключа *Key*.

В случае необходимости обеспечения защиты от повторов при передаче xDLMS APDU, защищенного на согласуемом сеансовом симметричном ключе, взаимодействующие стороны должны предусмотреть использование дополнительного механизма защиты. Такой механизм, к примеру, может базироваться на использовании меток времени и содержать этапы, приведенные в 7.4.2. Описание конкретного механизма защиты от повторов при передаче xDLMS APDU, защищенного на согласуемом сеансовом симметричном ключе, находится за рамками данного документа.

При реализации протокола DLMS в информационной системе, в которой актуальна угроза получения злоумышленником закрытого долговременного ключа, использование алгоритма согласования ключа с применением каждой из взаимодействующих сторон долговременных ключевых пар согласования ключа запрещается.

7.5 Механизмы аутентификации HLS

Аутентификация HLS (аутентификация высокого уровня безопасности) может осуществляться при установлении АА между клиентом и сервером.

В основе аутентификации HLS лежит схема из четырех этапов:

- этап 1: клиент отправляет серверу запрос, основу которого составляет случайное значение, сгенерированное клиентом;
- этап 2: сервер отправляет клиенту запрос, основу которого составляет случайное значение, сгенерированное сервером;
- этап 3: клиент вычисляет ответ на запрос сервера, обрабатывая этот запрос с применением криптографических алгоритмов и секретного значения, и отправляет полученный ответ серверу; сервер проверяет ответ клиента;
- этап 4: сервер вычисляет ответ на запрос клиента, обрабатывая этот запрос с применением криптографических алгоритмов и секретного значения, и отправляет полученный ответ клиенту; клиент проверяет ответ сервера.

Рассматриваются три механизма аутентификации HLS:

- механизм «HLS CMAC» на основе алгоритма вычисления имитовставки (см. 7.5.1);
- механизм «HLS GOST34112018-256» на основе алгоритма вычисления хэш-функции (см. 7.5.2);
- механизм «HLS GOST34102018-256» на основе алгоритма вычисления электронной цифровой подписи (см. 7.5.3).

Допускается применение

- механизма аутентификации «HLS CMAC» для согласования криптографического набора 8;
- механизмов аутентификации «HLS GOST34112018-256» и «HLS GOST34102018-256» для согласования криптографического набора 9.

Идентификаторы описываемых механизмов аутентификации HLS приведены в таблице 11. Содержимое запросов и ответов для каждого из описываемых механизмов аутентификации HLS приведено в таблице 12.

Т а б л и ц а 11 — Идентификаторы механизмов аутентификации HLS

Механизм аутентификации HLS	Идентификатор механизма (mechanism_id)	Полный идентификатор механизма аутентификации COSEM (COSEM authentication mechanism name)
«HLS CMAC»	mechanism_id(8)	2.16.756.5.8.2.8
«HLS GOST34112018-256»	mechanism_id(9)	2.16.756.5.8.2.9
«HLS GOST34102018-256»	mechanism_id(10)	2.16.756.5.8.2.10

Т а б л и ц а 12 — Содержимое запросов и ответов для механизмов аутентификации HLS

Механизм аутентификации HLS	Этап 1: запрос от клиента к серверу	Этап 2: запрос от сервера к клиенту	Этап 3: ответ от клиента к серверу	Этап 4: ответ от сервера к клиенту
mechanism_id(8) «HLS CMAC»	- случайная двоичная строка *CtoS* длины 64-512; - идентификатор клиента *system-title-C* длины 64 бит	- случайная двоичная строка *StoC* длины 64-512 бит; - идентификатор сервера *system-title-S* длины 64 бит	*SC* || IC_C || *KUZN_CMAC* ($LSB_{256}(K_{EM})$, IV_C || *SC* || *StoC* || *CtoS*)	*SC* || IC_S || *KUZN_CMAC* ($LSB_{256}(K_{EM})$, IV_S || *SC* || *CtoS* || *StoC*)
mechanism_id(9) «HLS GOST34112018-256»			$HASH_{256}$ (*HLS_Secret* || *system-title-C* || *system-title-S* || *StoC* || *CtoS*)	$HASH_{256}$ (*HLS_Secret* || *system-title-S* || *system-title-C* || *CtoS* || *StoC*)
mechanism_id(10) «HLS GOST34102018-256»	- случайная двоичная строка *CtoS* длины 256-512 бит; - идентификатор клиента *system-title-C* длины 64 бит; - сертификат ключа проверки подписи клиента (опционально)	- случайная двоичная строка *StoC* длины 256-512 бит; - идентификатор сервера *system-title-S* длины 64 бит; - сертификат ключа проверки подписи сервера (опционально)	$SIGN_{256}$ ($d_{sign,C}$, *system-title-C* || *system-title-S* || *StoC* || *CtoS*)	$SIGN_{256}$ ($d_{sign,S}$, *system-title-S* || *system-title-C* || *CtoS* || *StoC*)

Наличие строки *CtoS* и идентификатора *system-title-C* среди данных, передаваемых на этапе 1, а также строки *StoC* и идентификатора *system-title-S* среди данных, передаваемых на этапе 2, является обязательным для всех трех механизмов. В случае отсутствия этих значений среди передаваемых данных возвращается ошибка и AA не устанавливается.

Точные значения длин *CtoS* и *StoC* определяются конкретным исполнением протокола DLMS. При получении значения *StoC* на этапе 2 клиент должен проверить, что это значение не равно значению *CtoS*. В случае равенства значений *StoC* и *CtoS* возвращается ошибка и AA не устанавливается.

При получении значения *system-title-C* на этапе 1 сервер должен проверить, что это значение не равно значению *system-title-S*. Аналогично, при получении значения *system-title-S* на этапе 2 клиент

должен проверить, что это значение не равно значению *system-title-C*. В случае равенства значений *system-title-C* и *system-title-S* возвращается ошибка и AA не устанавливается.

Если стороны предварительно участвовали в процессе доверенной регистрации устройств с распределением идентификаторов (см. 6.2), то при получении идентификатора на этапе 1 и этапе 2 необходимо проверить его наличие среди идентификаторов, распределенных в процессе доверенной регистрации устройств. В случае, если полученный идентификатор отсутствует среди идентификаторов, распределенных в процессе доверенной регистрации устройств, возвращается ошибка и AA не устанавливается.

Для пересылки данных на этапе 1 используется AARQ, для чего устанавливаются следующие значения его параметров:

- значение запроса клиента:

calling-authentication-value = *CtoS*;

- идентификатор клиента:

calling-AP-title = *system-title-C*;

- уточняющие данные клиента (опционально): сертификат ключа проверки подписи клиента размещается в calling-AE-qualifier.

Перечисленные значения параметров пересылаются в незащищенном виде.

Для пересылки данных на этапе 2 используется AARE, для чего устанавливаются следующие значения его параметров:

- значение запроса сервера:

responding-authentication-value = *StoC*;

- идентификатор сервера:

responding-AP-title = *system-title-S*;

- уточняющие данные сервера (опционально): сертификат ключа проверки подписи сервера размещается в responding-AE-qualifier.

Перечисленные значения параметров пересылаются в незащищенном виде.

Для пересылки данных на этапе 3 и этапе 4 используется метод reply_to_HLS_authentication, для чего значение его параметра data устанавливается равным значению ответа на запрос. APDU, используемые в качестве транспорта для выполнения метода reply_to_HLS_authentication, при необходимости могут быть дополнительно зашифрованы и/или имитозащищены.

Аутентификация в каждом из трех механизмов считается успешной только в случае положительного результата всех проверок, осуществляемых клиентом и сервером и отсутствия сбоев и ошибок в процессе аутентификации. В случае успешной аутентификации устанавливается AA. Способы обработки сбоев и ошибок в процессе аутентификации HLS определяются конкретным исполнением протокола DLMS.

7.5.1 Механизм аутентификации HLS CMAC

Механизм аутентификации «HLS CMAC» может использоваться для согласования криптографического набора 8.

В качестве секрета, применяющегося при вычислении и проверке ответа в механизме аутентификации «HLS CMAC», используется глобальный ключ шифрования и имитозащиты при адресной рассылке $K_{EM} \in V_{512}$, известный только клиенту с идентификатором *system-title-C* и серверу с идентификатором *system-title-S*. Проверка ответа осуществляется путем вычисления значения имитовставки для идентичного набора данных и сравнения полученного и вычисленного значений имитовставки (предварительно должна быть выполнена проверка счетчика вызовов). В случае неравенства значений возвращается ошибка и AA не устанавливается.

Значения синхропосылок, использующихся при вычислении и проверке ответов в механизме аутентификации «HLS CMAC», имеют следующий вид:

$$IV_C = \textit{system-title-C} \,||\, IC_C \in V_{96},$$

$$IV_S = \textit{system-title-S} \,||\, IC_S \in V_{96},$$

где $IC_C \in V_{32}$ ($IC_S \in V_{32}$) — значение счетчика вызовов, уникальное для каждой операции зашифрования и/или вычисления имитовставки, выполняемой клиентом (сервером) с использованием ключа K_{EM}. Порядок инициализации, изменения и проверки значений счетчика вызовов описан в 6.3.

Байт управления безопасностью *SC*, использующийся при вычислении и проверке ответов в механизме аутентификации «HLS CMAC», должен принимать следующее значение:

$$SC = 0 \,||\, 0 \,||\, 0 \,||\, 1 \,||\, Vec_4(8).$$

7.5.2 Механизм аутентификации HLS GOST34112018-256

Механизм аутентификации «HLS GOST34112018-256» может использоваться для согласования криптографического набора 9.

В качестве секрета, применяющегося при вычислении и проверке ответа в механизме аутентификации «HLS GOST34112018-256», используется значение *HLS_Secret* $\in V^*$, известное только клиенту с идентификатором *system-title-C* и серверу с идентификатором *system-title-S*. Проверка ответа осуществляется путем вычисления значения хэш-функции для идентичного набора данных и сравнения полученного и вычисленного значений хэш-функции. В случае неравенства значений возвращается ошибка и AA не устанавливается.

Минимальная длина значения *HLS_Secret*, при использовании в механизме «HLS GOST34112018-256», должна составлять 128 бит. Начальные значения *HLS_Secret* устанавливаются на устройствах в процессе их изготовления. В дальнейшем возможно изменение значений *HLS_Secret* при помощи метода change_HLS_secret.

7.5.3 Механизм аутентификации HLS GOST34102018-256

Механизм аутентификации «HLS GOST34102018-256» может использоваться для согласования криптографического набора 9.

Используемые в механизме аутентификации «HLS GOST34102018-256» ключи подписи и проверки подписи должны соответствовать ГОСТ 34.10—2018 и эллиптической кривой, задаваемой набором параметров id-tc26-gost-3410-2012-256-paramSetB «1.2.643.7.1.2.1.1.2.», определенным в Р 1323565.1.024—2019.

В качестве секрета, применяющегося при вычислении ответа в механизме аутентификации «HLS GOST34102018-256» на этапе 3 (на этапе 4), используется ключ подписи клиента $d_{sign,C} \in \{1, 2,\ldots,q-1\}$ (ключ подписи сервера $d_{sign,S} \in \{1, 2,\ldots,q-1\}$). Проверка ответа осуществляется путем проверки полученного значения подписи с помощью соответствующего ключа проверки подписи (предварительно должна быть выполнена проверка самого ключа проверки подписи путем проверки соответствующего сертификата или цепочки сертификатов). В случае отрицательного результата хотя бы одной из проверок возвращается ошибка и AA не устанавливается.

Наличие сертификата ключа проверки подписи среди данных, передаваемых на этапе 1 и этапе 2 в механизме «HLS GOST34102018-256», является опциональным. При этом:

- если сертификат присутствует среди передаваемых данных, принимающая сторона должна использовать для проверки подписи этот сертификат;
- если сертификат не присутствует среди передаваемых данных, но был известен принимающей стороне, принимающая сторона должна использовать для проверки подписи имеющийся сертификат;
- если сертификат не присутствует среди передаваемых данных и не был известен принимающей стороне, возвращается ошибка и AA не устанавливается.

Приложение А
(справочное)

Контрольные примеры

Приводимые ниже значения рекомендуется использовать только для проверки корректной работы конкретной реализации алгоритмов, описанных в настоящих рекомендациях.

Все числовые значения приведены в шестнадцатеричной системе счисления.

В данном приложении двоичные строки из V^*, длина которых кратна 4, записываются в шестнадцатеричном виде, а символ конкатенации («||») опускается. То есть строка $a \in V_{4r}$ будет представлена в виде $a_{r-1}a_{r-2}\ldots a_0$, где $a_i \in \{0,1,\ldots,9,a,b,c,d,e,f\}$, $i = 0, 1, \ldots, r-1$. Соответствие между двоичными строками длины 4 и шестнадцатеричными строками длины 1 задается естественным образом (таблица А.1).

Преобразование, ставящее в соответствие двоичной строке длины $4r$ шестнадцатеричную строку длины r, и соответствующее обратное преобразование для простоты записи опускается.

Т а б л и ц а А.1 — Соответствие между двоичными и шестнадцатеричными строками

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	a
1011	b
1100	c
1101	d
1110	e
1111	f

П р и м е ч а н и е 1 — Далее по тексту символ «\\» обозначает перенос числа на новую строку.

П р и м е ч а н и е 2 — Во всех контрольных примерах, использующих функцию подписи $SIGN_{256}$, значение числа $k \in \{1, 2,\ldots,q-1\} \subset \mathbb{Z}_{2^{256}}$, применяющегося при формировании подписи согласно ГОСТ 34.10—2018, является фиксированным и задается как

k = 43730c5cbccacf915ac292676f21e8bd\\
4ef75331d9405e5f1a61dc3130a65011$_{16}$

А.1 Шифрование и имитозащита

Общие входные значения:

- $K_{EM} \in V_{512}$:
08090a0b0c0d0e0f0001020304050607\\
fedcba9876543210eca86420fdb97531\\
18191a1b1c1d1e1f1011121314151617\\
0123456789abcdef13579bdf02468ace$_{16}$

- *originator-system-title* $\in V_{64}$:
ff00ee11dd22cc33$_{16}$

- $IC_{EM} \in V_{32}$:
f0e1d2c3$_{16}$

- IV_{EM} = *originator-system-title* || $IC_{EM} \in V_{96}$:
ff00ee11dd22cc33f0e1d2c3$_{16}$

А.1.1 Случай $SC_5 = 0$, $SC_4 = 1$
Входные значения:

- $AAD \in V_{168}$:
188899aabbccddeeff00112233445566\\
$7789abcdef_{16}$

- $Plaintext \in V_0$:
null
Промежуточные значения:

- $K_M = LSB_{256}(K_{EM}) \in V_{256}$:
18191a1b1c1d1e1f1011121314151617\\
$0123456789abcdef13579bdf02468ace_{16}$
Выходные значения:

- $AuthTag = KUZN\_CMAC(K_M, IV_{EM} \| AAD) \in V_{96}$:
$8c5f4d499cd765c92c186216_{16}$

А.1.2 Случай $SC_5 = 1$, $SC_4 = 0$
Входные значения:

- $AAD \in V_0$:
null

- $Plaintext \in V_{160}$:
8899aabbccddeeff0011223344556677\\
$89abcdef_{16}$
Промежуточные значения:

- $K_E = MSB_{256}(K_{EM}) \in V_{256}$:
08090a0b0c0d0e0f0001020304050607\\
$fedcba9876543210eca86420fdb97531_{16}$
Выходные значения:

- $Ciphertext = KUZN\_CTR(K_E, IV_{EM}, Plaintext) \in V_{160}$:
7ec306cdc37237560df505975a629e7d\\
$012cfb11_{16}$

А.1.3 Случай $SC_5 = 1$, $SC_4 = 1$
Входные значения:

- $AAD \in V_8$:
38_{16}

- $Plaintext \in V_{160}$:
8899aabbccddeeff0011223344556677\\
$89abcdef_{16}$
Промежуточные значения:

- $K_E = MSB_{256}(K_{EM}) \in V_{256}$:
08090a0b0c0d0e0f0001020304050607\\
$fedcba9876543210eca86420fdb97531_{16}$

- $K_M = LSB_{256}(K_{EM}) \in V_{256}$:
18191a1b1c1d1e1f1011121314151617\\
$0123456789abcdef13579bdf02468ace_{16}$
Выходные значения:

- $Ciphertext = KUZN\_CTR(K_E, IV_{EM}, Plaintext) \in V_{160}$:
7ec306cdc37237560df505975a629e7d\\
$012cfb11_{16}$

- $AuthTag = KUZN\_CMAC(K_M, IV_{EM} \| AAD \| Ciphertext) \in V_{96}$:
$21206647b4840b0d75537098_{16}$

А.2 Передача ключа

А.2.1 Вычисление экспортного представления ключа
Входные значения:

- $K_{KEK} \in V_{512}$:
08090a0b0c0d0e0f0001020304050607\\

fedcba9876543210eca86420fdb97531\\
18191a1b1c1d1e1f1011121314151617\\
0123456789abcdef13579bdf02468ace$_{16}$

- *system-title-U* $\in V_{64}$:
ff00ee11dd22cc33$_{16}$

- $IC_{KEK} \in V_{32}$:
f0e1d2c3$_{16}$

- *Key* $\in V_{512}$:
28292a2b2c2d2e2f2021222324252627\\
ffeeddccbbaa99880011223344556677\\
38393a3b3c3d3e3f3031323334353637\\
ffffeeeeddddcccc0000111122223333$_{16}$
Промежуточные значения:

- IV_{KEK} = *system-title-U* || $IC_{KEK} \in V_{96}$:
ff00ee11dd22cc33f0e1d2c3$_{16}$
- $K_{KEKE} = MSB_{256}(K_{KEK}) \in V_{256}$:
08090a0b0c0d0e0f0001020304050607\\
fedcba9876543210eca86420fdb97531$_{16}$

- $K_{KEKM} = LSB_{256}(K_{KEK}) \in V_{256}$:
18191a1b1c1d1e1f1011121314151617\\
0123456789abcdef13579bdf02468ace$_{16}$
Выходные значения:

- $ExpKey = KUZN\_KEXP(Key,K_{KEKM},K_{KEKE},IV_{KEK}) \in V_{640}$:
de73865d2382f7862dc505873a12de2d\\
7769eb32fa9b706885956053ec7e7a17\\
0e96a817dd51f0d47a5f211914c79c40\\
54538cf43dd539a58e3c555e603ac9be\\
a071555e3e832b377c670f997ad7faa6$_{16}$

А.3 Защита данных с помощью электронной цифровой подписи

Входные значения:

- $d_{sign} \in \{1, 2,\ldots,q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{sign}) \in V_{256}$:
48494a4b4c4d4e4f4041424344454647\\
bbbbaaaa999988884444555566667777$_{16}$

- $AF \in V_{64}$:
7700661155224433$_{16}$

- *Content* $\in V_{160}$:
8899aabbccddeeff0011223344556677\\
89abcdef$_{16}$
Промежуточные значения:

- *SignData* = *AF* || *Content* $\in V_{224}$:
770066115522443388 99aabbccddeeff\\
00112233445566778 89abcdef$_{16}$
Выходные значения:

- *Signature* = $SIGN_{256}(d_{sign}, SignData) \in V_{512}$:
d3b72bb12fb7da1a06f8e11acdec034f\\
fcf14588301a3315bbe8cd611fc4545e\\
a9fae88aeac47cd46a0858711d942223\\
c523bfd53cbadff97e0eec1f69a3efca$_{16}$

А.4 Согласование ключа

А.4.1 Согласование ключа с применением каждой из взаимодействующих сторон сеансовых ключевых пар согласования ключа

Входные значения:

- $d_{s,U}^{sign} \in \{1, 2,\ldots, q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Ves_{256}(d_{s,U}^{sign}) \in V_{256}(d_{s,U}^{sign}) \in V_{256}$:
48494a4b4c4d4e4f4041424344454647\\
bbbbaaaa999988884444555566667777$_{16}$

- $Q_{s,U}^{sign} \in E$, где $\pi_x(Q_{s,U}^{sign}) \,||\, (\pi_y(Q_{s,U}^{sign}) \in F_{512}$:
4317f72b8458cb1b76d6cb9191ae19f1\\
ec202b243a4c3cb8975d6f395e6cf397\\
9de7576fd6d00dcd66902ea7bc3bf9ca\\
0c017e010228e81b07736485259e2e08$_{16}$

- $d_{s,V}^{sign} \in \{1, 2,\ldots, q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{s,V}^{sign}) \in V_{256}$:
58595a5b5c5d5e5f5051525354555657\\
ffffffffeeeeeeee8888888899999999$_{16}$

- $Q_{s,V}^{sign} \in E$, где $\pi_x(Q_{s,V}^{sign}) \,||\, \pi_y(Q_{s,V}^{sign}) \in V_{512}$:
4af7b22315a787b871071907acd69016\\
620706763d1b9bcab3ae7b8bae25e100\\
3be835637450a50cf8a45293e4553d8d\\
21f7d3c6cc81e7445053031c8ab93f52$_{16}$

- $key\_id \in V_{16}$:
1600$_{16}$

- $AlgorithmID \in V_{56}$:
60857406080304$_{16}$

- $system\text{-}title\text{-}U \in V_{64}$:
ff00ee11dd22cc33$_{16}$

- $system\text{-}title\text{-}V \in V_{64}$:
bb44aa5599668877$_{16}$

Промежуточные значения:

- $d_{e,U}^{agr} \in \{1,2,\ldots, q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{e,U}^{agr}) \in V_{256}$:
68696a6b6c6d6e6f6061626364656667\\
Ddddddddccccccccaaaaaaaabbbbbbbb$_{16}$

- $Q_{e,U}^{agr} \in E$, где $\pi_x(Q_{e,U}^{agr}) \,||\, \pi_y(Q_{e,U}^{agr}) \in V_{512}$:
95da164bcee759b2ae0f1860c9845d34\\
734b5ae066aeaa3fcf4394bfec09d4d3\\
f4a226a0015ca8c9338ea12dbe83502a\\
581ecc15b80ace45c7f25bc8275962a7$_{16}$

- $d_{e,V}^{agr} \in \{1,2,\ldots, q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{e,V}^{agr}) \in V_{256}$;
78797a7b7c7d7e7f7071727374757677\\
77777777666666660000000011111111$_{16}$

- $Q_{e,V}^{agr} \in E$, где $\pi_x(Q_{e,V}^{agr}) \,||\, \pi_y(Q_{e,V}^{agr}) \in V_{512}$:
212daf02de1c91ea961e58e01e42df17\\
33c00748998bc34d76dad96b3b256378\\
7b9cffcfa0f24753d6d5eb6133b35a95\\
375a0ef683b3ff5be7d61b99d7fe6617$_{16}$

- $P_{VU} = VKO_{256}(d_{e,U}^{agr}, Q_{e,V}^{agr}, 1) = VKO_{256}(d_{e,V}^{agr}, Q_{e,U}^{agr}, 1) = P_{UV} \in V_{256}$:
0f4af3ef046cf27a5c9a83616e530b0b\\
d9ade5373cd2b31912ef1fb67dad0916$_{16}$

- $T_{VU} = KDFTREE(P_{VU}, AlgorithmID, system\text{-}title\text{-}U \,||\, system\text{-}title\text{-}V, 1) =$
$= KDFTREE(P_{UV}, AlgorithmID, system\text{-}title\text{-}U \,||\, system\text{-}title\text{-}V, 1) = T_{UV} \in V_{768}$:
ab280aed3b9279b34ff03009574df9ab\\
9d053f724c9d7ea9a534a6d3bc4bc660\\
a377ff93d5f5fab34f0ee2173f1303d2\\
2f762b2991f741c76943021f92f86235\\
2282eab5b6c7d7ed36635277a4ad1292\\
96751f690a4d8ade1adcd2988eb9ea77$_{16}$

- $M_{VU} = MSB_{256}(T_{VU}) = MSB_{256}(T_{UV}) = M_{UV} \in V_{256}$:
ab280aed3b9279b34ff03009574df9ab\\
$9d053f724c9d7ea9a534a6d3bc4bc660_{16}$

- $SignData_U = key\_id \| \pi_x(Q_{e,U}^{agr}) \| \pi_x(Q_{e,V}^{agr}) \| system\text{-}title\text{-}V \in V_{256}$:
160095da164bcee759b2ae0f1860c984\\
5d34734b5ae066aeaa3fcf4394bfec09\\
d4d3212daf02de1c91ea961e58e01e42\\
df1733c00748998bc34d76dad96b3b25\\
$6378bb44aa5599668877_{16}$

- $sign_U = SIGN_{256}(d_{s,U}^{sign}, SignData_U) \in V_{512}$:
d3b72bb12fb7da1a06f8e11acdec034f\\
fcf14588301a3315bbe8cd611fc4545e\\
c6c972bb580b7e8df39a3403796a61cc\\
$a3c2b037c49440ea812928f0f45d9ba4_{16}$

- $TagData_U = \pi_x(Q_{e,U}^{agr}) \| \pi_x(Q_{e,V}^{agr}) \| system\text{-}title\text{-}U \| system\text{-}title\text{-}V \in V_{640}$:
95da164bcee759b2ae0f1860c9845d34\\
734b5ae066aeaa3fcf4394bfec09d4d3\\
212daf02de1c91ea961e58e01e42df17\\
33c00748998bc34d76dad96b3b256378\\
$ff00ee11dd22cc33bb44aa5599668877_{16}$

- $tag_U = KUZN\_CMAC(M_{VU}, TagData_U) \in V_{96}$:
$64d57e57bcfb420d596d5303_{16}$

- $SignData_V = \pi_x(Q_{e,V}^{agr}) \| \pi_x(Q_{e,U}^{agr}) \| system\text{-}title\text{-}U \in V_{576}$:
212daf02de1c91ea961e58e01e42df17\\
33c00748998bc34d76dad96b3b256378\\
95da164bcee759b2ae0f1860c9845d34\\
734b5ae066aeaa3fcf4394bfec09d4d3\\
$ff00ee11dd22cc33_{16}$

- $sign_V = SIGN_{256}(d_{s,V}^{sign}, SignData_V) \in V_{512}$:
d3b72bb12fb7da1a06f8e11acdec034f\\
fcf14588301a3315bbe8cd611fc4545e\\
dc05dedeb043e340a503172c298fb58a\\
$4e8fe28a0fa3963a432c64f87f5e9761_{16}$

- $TagData_V = \pi_x(Q_{e,V}^{agr}) \| \pi_x(Q_{e,U}^{agr}) \| system\text{-}title\text{-}V \| system\text{-}title\text{-}U \in V_{640}$:
212daf02de1c91ea961e58e01e42df17\\
33c00748998bc34d76dad96b3b256378\\
95da164bcee759b2ae0f1860c9845d34\\
734b5ae066aeaa3fcf4394bfec09d4d3\\
$bb44aa5599668877ff00ee11dd22cc33_{16}$

- $tag_V = KUZN\_CMAC(M_{UV}, TagData_V) \in V_{96}$:
$978c1437c532b922e30fdf19_{16}$

Выходные значения:

- $K_{VU} = LSB_{512}(T_{VU}) = LSB_{512}(T_{UV}) = K_{UV} \in V_{512}$:
a377ff93d5f5fab34f0ee2173f1303d2\\
2f762b2991f741c76943021f92f86235\\
2282eab5b6c7d7ed36635277a4ad1292\\
$96751f690a4d8ade1adcd2988eb9ea77_{16}$

А.4.2 Согласование ключа с применением одной из взаимодействующих сторон сеансовой ключевой пары согласования ключа и другой из сторон долговременной ключевой пары согласования ключа

Входные значения:

- $d_{s,U}^{sign} \in \{1, 2, \ldots, q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{s,U}^{sign}) \in V_{256}$;
48494a4b4c4d4e4f4041424344454647\\
$bbbbaaaa99998888444455556666777 7_{16}$

- $Q_{s,U}^{sign} \in E$, где $\pi_x(Q_{s,U}^{sign}) \| \pi_y(Q_{s,U}^{sign}) \in V_{512}$:
4317f72b8458cb1b76d6cb9191ae19f1\\

ec202b243a4c3cb8975d6f395e6cf397\\
9de7576fd6d00dcd66902ea7bc3bf9ca\\
0c017e010228e81b07736485259e2e08$_{16}$

- $d_{s,V}^{agr} \in \{1, 2,\ldots, q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{s,V}^{agr}) \in V_{256}$;
78797a7b7c7d7e7f7071727374757677\\
77777777666666660000000011111111$_{16}$

- $Q_{s,V}^{agr} \in E$, где $\pi_x(Q_{s,V}^{agr}) \,||\, \pi_y(Q_{s,V}^{agr}) \in V_{512}$:
212daf02de1c91ea961e58e01e42df17\\
33c00748998bc34d76dad96b3b256378\\
7b9cffcfa0f24753d6d5eb6133b35a95\\
375a0ef683b3ff5be7d61b99d7fe6617$_{16}$

- $AlgorithmID \in V_{56}$:
60857406080304$_{16}$

- $system\text{-}title\text{-}U \in V_{64}$:
ff00ee11dd22cc33$_{16}$

- $system\text{-}title\text{-}V \in V_{64}$:
bb44aa5599668877$_{16}$

Промежуточные значения:

- $r_U \in V_{128}$:
f0f0f0f0e1e1e1e1d2d2d2d2c3c3c3c3$_{16}$

- $d_{e,U}^{agr} \in \{1, 2,\ldots, q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{e,U}{}^{agr}) \in V_{256}$;
68696a6b6c6d6e6f60616263646566 67\\
Ddddddddccccccccaaaaaaaabbbbbbbb$_{16}$

- $Q_{e,U}^{agr} \in E$, где $\pi_x(Q_{e,U}^{agr}) \,||\, \pi_y(Q_{e,U}^{agr}) \in V_{512}$:
95da164bcee759b2ae0f1860c9845d34\\
734b5ae066aeaa3fcf4394bfec09d4d3\\
f4a226a0015ca8c9338ea12dbe83502a\\
581ecc15b80ace45c7f25bc8275962a7$_{16}$

- $P_{VU} = VKO_{256}(d_{e,U}^{agrr}, Q_{s,V}^{agr}, r_U) = VKO_{256}(d_{s,V}^{agr}, Q_{e,U}^{agr}, r_U) = P_{UV} \in V_{256}$:
4f54f663029709c0271facd5bb6d5818\\
7410477e102555a893d45a04ab0cafc0$_{16}$

- $T_{VU} = KDFTREE(P_{VU}, AlgorithmID, system\text{-}title\text{-}U \,||\, system\text{-}title\text{-}V, 1) =$
$= KDFTREE(P_{UV}, AlgorithmID, system\text{-}title\text{-}U \,||\, system\text{-}title\text{-}V, 1) = T_{UV} \in V_{768}$:
e7f74dc8fcafd9738fd14d5aa542834b\\
ac7e883eff37931c082a9a80b45f60dd\\
159d1118b56f8e78e938c28715c34c3c\\
197a2339638901de1c610180f7de34ac\\
424237f626e9ae5b55dbfa12ffd9cb7d\\
fb903019eecc8228876015b2c15cbc89$_{16}$

- $M_{VU} = MSB_{256}(T_{VU}) = MSB_{256}(T_{UV}) = M_{UV} \in V_{256}$:
e7f74dc8fcafd9738fd14d5aa542834b\\
ac7e883eff37931c082a9a80b45f60dd$_{16}$

- $SignData_U = \pi_x(Q_{e,U}^{agr}) \,||\, \pi_x(Q_{s,V}^{agr}) \,||\, system\text{-}title\text{-}V \in V_{576}$:
95da164bcee759b2ae0f1860c9845d34\\
734b5ae066aeaa3fcf4394bfec09d4d3\\
212daf02de1c91ea961e58e01e42df17\\
33c00748998bc34d76dad96b3b256378\\
bb44aa5599668877$_{16}$

- $sign_U = SIGN_{256}(d_{s,U}^{sign}, SignData_U) \in V_{512}$:
d3b72bb12fb7da1a06f8e11acdec034f\\
fcf14588301a3315bbe8cd611fc4545e\\
0e6bf8801cfc4c6514da487d46f100e3\\
607a58ed91e33d183cc04b739c983d00$_{16}$

- $TagData_U = r_U \| \pi_x(Q_{e,U}^{agr}) \| \pi_x(Q_{s,V}^{agr}) \| system\text{-}title\text{-}U \| system\text{-}title\text{-}V \in V_{768}$:
f0f0f0f0e1e1e1e1d2d2d2d2c3c3c3c3\\
95da164bcee759b2ae0f1860c9845d34\\
734b5ae066aeaa3fcf4394bfec09d4d3\\
212daf02de1c91ea961e58e01e42df17\\
33c00748998bc34d76dad96b3b256378\\
$ff00ee11dd22cc33bb44aa5599668877_{16}$

- $tag_U = KUZN\_CMAC(M_{VU}, TagData_U) \in V_{96}$:
$ce3e8b5e1f70cd32d70c74be_{16}$
Выходные значения:

- $K_{VU} = LSB_{512}(T_{VU}) = LSB_{512}(T_{UV}) = K_{UV} \in V_{512}$:
159d1118b56f8e78e938c28715c34c3c\\
197a2339638901de1c610180f7de34ac\\
424237f626e9ae5b55dbfa12ffd9cb7d\\
$fb903019eecc8228876015b2c15cbc89_{16}$

A.5 Механизмы аутентификации HLS

A.5.1 HLS CMAC

Входные значения:

- $K_{EM} \in V_{512}$:
08090a0b0c0d0e0f0001020304050607\\
fedcba9876543210eca86420fdb97531\\
18191a1b1c1d1e1f1011121314151617\\
$0123456789abcdef13579bdf02468ace_{16}$

- $system\text{-}title\text{-}C \in V_{64}$:
$ff00ee11dd22cc33_{16}$

- $IC_C \in V_{32}$:
$f0e1d2c3_{16}$

- $system\text{-}title\text{-}S \in V_{64}$:
$bb44aa5599668877_{16}$

- $IC_S \in V_{32}$:
01234567_{16}

- $SC \in V_8$:
18_{16}

- $StoC \in V_{64}$:
$8899aabbccddeeff_{16}$

- $CtoS \in V_{64}$:
0011223344556677_{16}
Промежуточные значения:

- $IV_C = system\text{-}title\text{-}C \| IC_C \in V_{96}$:
$ff00ee11dd22cc33f0e1d2c3_{16}$

- $IV_S = system\text{-}title\text{-}S \| IC_S \in V_{96}$:
$bb44aa559966887701234567_{16}$

- $K_M = LSB_{256}(K_{EM}) \in V_{256}$:
18191a1b1c1d1e1f1011121314151617\\
$0123456789abcdef13579bdf02468ace_{16}$

- $TagData_C = IV_C \| SC \| StoC \| CtoS \in V_{232}$:
ff00ee11dd22cc33f0e1d2c3188899aa\\
$bbccddeeff0011223344556677_{16}$

- $AuthTag_C = KUZN\_CMAC(K_M, TagData_C) \in V_{96}$:
$2b0a59e54ab716489fcfc7f6_{16}$

- $TagData_S = IV_S \| SC \| CtoS \| StoC \in V_{232}$:
bb44aa559966887701234567180011223
$33445566778899aabbccddeeff_{16}$

- $AuthTag_S = KUZN\_CMAC(K_M, TagData_S) \in V_{96}$:
$f3406929fbcd40b532a07793_{16}$
Выходные значения:

- $Answer_C = SC \parallel IC_C \parallel AuthTag_C \in V_{136}$:
$18f0e1d2c32b0a59e54ab716489fcfc7\backslash\backslash$
$f6_{16}$

- $Answer_S = SC \parallel IC_S \parallel AuthTag_S \in V_{136}$:
$1801234567f3406929fbcd40b532a077\backslash\backslash$
93_{16}

A.5.2 HLS GOST34112018-256
Входные значения:

- $HLS\_Secret \in V_{256}$:
$78797a7b7c7d7e7f7071727374757677\backslash\backslash$
$88898a8b8c8d8e8f8081828384858687_{16}$

- $system\text{-}title\text{-}C \in V_{64}$:
$ff00ee11dd22cc33_{16}$

- $system\text{-}title\text{-}S \in V_{64}$:
$bb44aa5599668877_{16}$

- $StoC \in V_{64}$:
$8899aabbccddeeff_{16}$

- $CtoS \in V_{64}$:
0011223344556677_{16}
Промежуточные значения:

- $HashData_C = HLS\_Secret \parallel system\text{-}title\text{-}C \parallel system\text{-}title\text{-}S \parallel StoC \parallel CtoS \in V_{512}$:
$78797a7b7c7d7e7f7071727374757677\backslash\backslash$
$88898a8b8c8d8e8f8081828384858687\backslash\backslash$
$ff00ee11dd22cc33bb44aa5599668877\backslash\backslash$
$8899aabbccddeeff0011223344556677_{16}$

- $HashData_S = HLS\_Secret \parallel system\text{-}title\text{-}S \parallel system\text{-}title\text{-}C \parallel CtoS \parallel StoC \in V_{512}$:
$78797a7b7c7d7e7f7071727374757677\backslash\backslash$
$88898a8b8c8d8e8f8081828384858687\backslash\backslash$
$bb44aa5599668877ff00ee11dd22cc33\backslash\backslash$
$00112233445566778899aabbccddeeff_{16}$
Выходные значения:

- $Answer_C = HASH_{256}(HashData_C) \in V_{256}$:
$4c375b843898b6f0a0744051f74e42f2\backslash\backslash$
$a944581d46c495e743e97abdcd9d7c58_{16}$

- $Answer_S = HASH_{256}(HashData_S) \in V_{256}$:
$55dcd7e597cc90ec215c2faae8f86c0a\backslash\backslash$
$1d707ddeac1adf5cbd17dfaa5378c500_{16}$

A.5.3 HLS GOST34102018-256
Входные значения:

- $d_{sign,C} \in \{1, 2,\ldots,q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{sign,C}) \in V_{256}$:
$48494a4b4c4d4e4f4041424344454647\backslash\backslash$
$bbbbaaaa999988884444555566667777_{16}$

- $d_{sign,S} \in \{1, 2,\ldots,q-1\} \subset \mathbb{Z}_{2^{256}}$, где $Vec_{256}(d_{sign,S}) \in V_{256}$:
$58595a5b5c5d5e5f5051525354555657\backslash\backslash$
$ffffffffeeeeeeee8888888899999999_{16}$

- $system\text{-}title\text{-}C \in V_{64}$:
$ff00ee11dd22cc33_{16}$

- $system\text{-}title\text{-}S \in V_{64}$:
$bb44aa5599668877_{16}$

- $StoC \in V_{256}$:
$8899aabbccddeeff88889999aaaabbbb\backslash\backslash$
$ccccddddeeeeffff89abcdeffedcba98_{16}$

- $CtoS \in V_{256}$:
00112233445566770000111122223333\\
44445555666677770123456776543210$_{16}$
Промежуточные значения:

- $SignData_C$ = *system-title-C* || *system-title-S* || *StoC* || $CtoS \in V_{640}$:
ff00ee11dd22cc33bb44aa5599668877\\
8899aabbccddeeff88889999aaaabbbb\\
ccccddddeeeeffff89abcdeffedcba98\\
00112233445566770000111122223333\\
44445555666677770123456776543210$_{16}$

- $SignData_S$ = *system-title-S* || *system-title-C* || *CtoS* || *StoC* $\in V_{640}$:
bb44aa5599668877ff00ee11dd22cc33\\
00112233445566770000111122223333\\
44445555666677770123456776543210\\
8899aabbccddeeff88889999aaaabbbb\\
ccccddddeeeeffff89abcdeffedcba98$_{16}$
Выходные значения:

- $Answer_C = SIGN_{256}(d_{sign,C}, SignData_C) \in V_{512}$:
d3b72bb12fb7da1a06f8e11acdec034f\\
fcf14588301a3315bbe8cd611fc4545e\\
38d6bfaa0b2b03bc17afbef8b8fe41c1\\
38bba0308f3ae8c98b07f968776b861b

- $Answer_S = SIGN_{256}(d_{sign,S}, SignData_S) \in V_{512}$:
d3b72bb12fb7da1a06f8e11acdec034f\\
fcf14588301a3315bbe8cd611fc4545e\\
dfdd24c387320ae2e5109f4e32851d23\\
c3219de99826a844ad8879f54800dc00$_{16}$

Библиография

[1] DLMS/COSEM Architecture and Protocols, Edition 8.0.
[2] COSEM Interface Classes and OBIS Object Identification System, Edition 12.0.

УДК 681.3.06:006.354 ОКС 35. 040

Ключевые слова: криптографические протоколы, шифрование, аутентификация, ключ, имитозащита

Редактор *П.К. Одинцов*
Технический редактор *В.Н. Прусакова*
Корректор *О.В. Лазарева*
Компьютерная верстка *Е.О. Асташина*

Сдано в набор 28.10.2020. Подписано в печать 04.12.2020. Формат 60×84 1/8. Гарнитура Ариал.
Усл. печ. л. 4,65. Уч.-изд. л. 4,18.
Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Создано в единичном исполнении во ФГУП «СТАНДАРТИНФОРМ» для комплектования Федерального информационного фонда стандартов, 117418 Москва, Нахимовский пр-т, д. 31, к. 2.
www.gostinfo.ru info@gostinfo.ru