ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

ПРЕДВАРИТЕЛЬНЫЙ НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПНСТ 820— 2023

Информационные технологии

ИНТЕРНЕТ ВЕЩЕЙ

Протокол передачи данных для высокоемких сетей на основе сверхузкополосной модуляции радиосигнала (OpenUNB)

Издание официальное

Москва
Российский институт стандартизации
2023

Предисловие

1 РАЗРАБОТАН центром компетенций НТИ «Сквозные технологии беспроводной связи и интернета вещей» АНОО ВПО «Сколковский институт науки и технологий»

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 194 «Кибер-физические системы»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 13 марта 2023 г. № 15-пнст

Правила применения настоящего стандарта и проведения его мониторинга установлены в ГОСТ Р 1.16—2011 (разделы 5 и 6).

Федеральное агентство по техническому регулированию и метрологии собирает сведения о практическом применении настоящего стандарта. Данные сведения, а также замечания и предложения по содержанию стандарта можно направить не позднее чем за 4 мес до истечения срока его действия разработчику настоящего стандарта по адресу: 121205 Москва, Инновационный Центр Сколково, Большой бульвар, д. 30, стр. 1 и/или в Федеральное агентство по техническому регулированию и метрологии: 123112 Москва, Пресненская набережная, д. 10, стр. 2.

В случае отмены настоящего стандарта соответствующая информация будет опубликована в ежемесячном информационном указателе «Национальные стандарты» и также будет размещена на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.rst.gov.ru)

Содержание

Введение

Настоящий стандарт определяет протокол односторонней передачи данных от множества конечных устройств до центрального сервера сети. Протокол оптимизирован для мобильных и стационарных конечных устройств с батарейным питанием.

Сети, построенные в соответствии с протоколом OpenUNB, относятся к классу LPWAN (Low-power Wide-area Network) — беспроводных энергоэффективных сетей дальнего радиуса действия. Сеть OpenUNB имеет топологию «звезда из звезд», в которой шлюзы ретранслируют пакеты данных, принятые от конечных устройств, на единый сервер сети. Сервер сети в свою очередь передает успешно принятые пакеты конечных устройств на соответствующий сервер приложений.

Протокол OpenUNB использует симметричное шифрование для обеспечения конфиденциальности, целостности и подлинности данных, передаваемых конечными устройствами. Шифрование осуществляется на конечных устройствах с использованием сессионных ключей, которые вырабатываются из секретного ключа устройства, а расшифрование осуществляется на сервере сети.

Связь между конечными устройствами и шлюзами осуществляется с использованием сверхузкополосных сигналов с модуляцией DBPSK (Differential Binary Phase Shift Keying) или FSK (Frequency Shift Keying) и помехоустойчивого полярного кодирования. Это позволяет добиться высокой дальности, энергоэффективности и емкости (пропускной способности) сети.

Таким образом, протокол OpenUNB является эффективным для построения телеметрических систем с большим количеством конечных устройств с батарейным питанием и может быть использован в нелицензируемых диапазонах частот, имеющих ограничение на мощность и скважность излучаемого сигнала.

ПРЕДВАРИТЕЛЬНЫЙ НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационные технологии

ИНТЕРНЕТ ВЕЩЕЙ

Протокол передачи данных для высокоемких сетей на основе сверхузкополосной модуляции радиосигнала (OpenUNB)

Information technology. Internet of things. Wireless protocol based on ultra narrow band RF modulation (OpenUNB)

Срок действия — с 2023—03—31
до 2026—03—31

1 Область применения

Сети OpenUNB, построенные в соответствии с настоящим стандартом, относятся к классу энергоэффективных сетей дальнего радиуса действия или Low Power Wide Area Network (LPWAN).

Протокол OpenUNB предназначен для односторонней передачи данных от множества конечных устройств (например, датчиков и счетчиков) до центрального сервера сети с ретрансляцией через множество шлюзов. Связь между конечными устройствами и шлюзами осуществляется по радиоканалу с использованием сверхузкополосных сигналов с модуляцией DBPSK (Differential Binary Phase Shift Keying) или FSK (Frequency Shift Keying) и помехоустойчивого полярного кодирования. Это позволяет добиться высокой дальности и энергоэффективности передачи и высокой емкости (пропускной способности) сети.

Настоящий стандарт предназначен для построения телеметрических систем с большим количеством конечных устройств, имеющих батарейное питание, в которых требуется обеспечить максимально долгую автономную работу устройств (без обслуживания и замены элементов питания) и передачу данных на большие расстояния.

Настоящий стандарт реализует криптографическую защиту передаваемых данных, обеспечивающую их конфиденциальность, целостность и подлинность.

Высокая энергоэффективность передачи данных позволяет применять настоящий стандарт для построения сетей, работающих в нелицензируемых диапазонах частот, имеющих ограничения на мощность излучаемого передатчиком сигнала, например в диапазоне 868.7—869.2 МГц.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

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

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

Примечание — При пользовании настоящим стандартом 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 стандарт, на который

Издание официальное

дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1 **(конечное) устройство:** Программно-аппаратный комплекс, предназначенный для передачи пакетов данных на сервер.

3.2 **сеансовый ключ:** Временный ключ, синхронно вырабатываемый на конечном устройстве и на сервере сети и имеющий время жизни, равное одной эпохе работы конечного устройства.

3.3 **сервер сети:** Программно-аппаратный комплекс, предназначенный для приема пакетов данных конечных устройств от шлюзов и их последующей передачи на сервер приложений.

3.4 **сервер приложений:** Программно-аппаратный комплекс, предназначенный для приема пакетов данных конечных устройств от сервера сети, их хранения и отображения конечным пользователям.

3.5 **шлюз:** Программно-аппаратный комплекс, предназначенный для приема пакетов данных конечных устройств из радиоканала и их последующей передачи на сервер сети.

3.6 **эпоха:** Ограниченный период времени, в течение которого конечное устройство передает пакеты данных серверу сети.

4 Обозначения и сокращения

В настоящем стандарте применены следующие обозначения и сокращения:

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

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

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

$A||B$ — конкатенация двоичных строк $A, B \in V^*$, т. е. строка из $V_{|A|+|B|}$, в которой подстрока с большими номерами компонент из $V_{|A|}$ совпадает со строкой A, а подстрока с меньшими номерами компонент из $V_{|B|}$ совпадает со строкой B;

0^r — строка, состоящая из r нулей;

$MSB_s(Z)$ — усечение битовой строки Z путем взятия s старших бит: строке $z_{m-1}|| \ldots \ldots ||z_1||z_0$, $m \geq s$, ставится в соответствие строка $z_{m-1}|| \ldots ||z_{m-s+1}||z_{m-s}$, $z_i \in V_1$, $i = 0, 1, \ldots, m - 1$;

k — параметр алгоритма блочного шифрования, называемый длиной ключа;

CTR (K, IV, P) — режим гаммирования, определяемый ГОСТ Р 34.13, с параметром s = 64. Здесь K — ключ шифрования, IV — синхропосылка, $P \in V^*$ — шифруемые (расшифруемые) данные;

ECB (K, P) — режим простой замены, определяемый ГОСТ Р 34.13. Здесь K — ключ шифрования, $P \in V^*$ — шифруемые (расшифруемые) данные;

$CMAC_{24}$ (K, P) — режим выработки имитовставки, определяемый ГОСТ Р 34.13, с параметром s = 24. Здесь K — ключ шифрования, $P \in V^*$ — данные, для которых требуется вычислить имитовставку;

CRC24 — циклический избыточный код длиной 24 бита;

DevID — идентификатор конечного устройства;

DevAddr — временный адрес конечного устройства;

K_0 — секретный ключ шифрования конечного устройства;

K_e — сеансовый ключ шифрования;

K_m — сеансовый ключ расчета имитовставки;

N_a — номер активации конечного устройства;

N_e — номер текущей эпохи устройства от момента его активации;
N_n — номер пакета, передаваемого конечным устройством;
DBPSK — дифференциальная (относительная) двоичная фазовая манипуляция несущей (Differential Binary Phase Shift Keying);
FSK — двоичная частотная манипуляция несущей (Frequency Shift Keying);
LLC — подуровень управления логической связью (Logical Link Control);
LPWAN — энергоэффективная сеть дальнего радиуса действия (Low Power Wide Area Network);
MAC — подуровень управления доступом к среде (Medium Access Control);
MIC — имитовставка или код целостности сообщения (Message Integrity Code);
PHY — физический уровень (Physical layer);
UNB — сверхузкополосный сигнал (Ultra Narrow Band).

В настоящем стандарте используются следующие форматы записей:
- константы записываются в формате CONST_PARAM_NAME;
- названия полей в форматах передаваемых пакетов записываются в формате PacketFieldName;
- названия временных переменных в процедурах записываются в формате variable_name.

5 Архитектура сети OpenUNB

5.1 Общая архитектура

На рисунке 1 представлена общая архитектура сети.

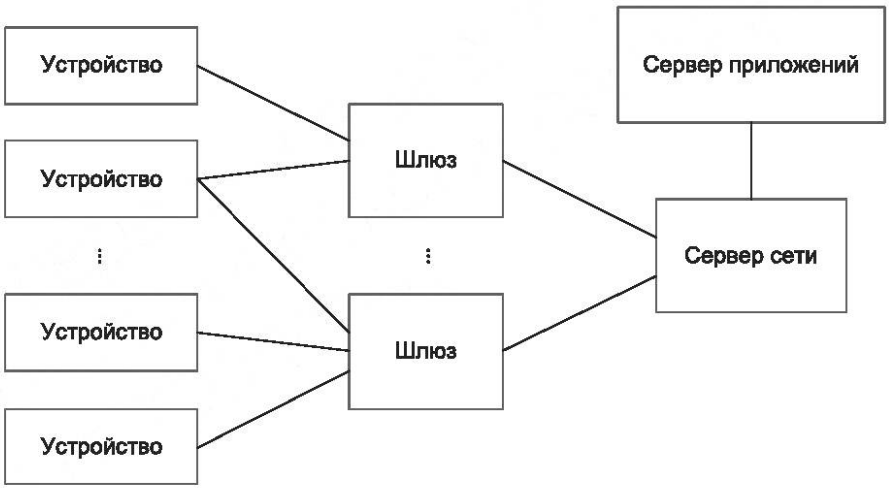

Рисунок 1 — Архитектура сети

Конечные устройства передают пакеты данных по радиоканалу в соответствии с настоящим стандартом. Передача пакетов является широковещательной. Получателями пакетов являются шлюзы, находящиеся в зоне радиовидимости устройства.

Архитектура сети OpenUNB удовлетворяет требованиям международных стандартов, описывающих архитектуру беспроводных сетей LPWAN со сверхузкополосной модуляцией (см., например, [1]). Сеть OpenUNB предусматривает три уровня сетевого взаимодействия:
- физический уровень;
- канальный уровень;
- уровень приложений.

5.2 Функции физического уровня

Физический уровень протокола определяет механизм приема-передачи пакетов канального уровня по радиоканалу, в том числе формат пакетов. Передача пакетов является односторонней. Передатчик физического уровня реализуется на конечных устройствах, а приемник — на шлюзах сети.

Функции конечных устройств:
- помехоустойчивое кодирование данных канального уровня;
- формирование пакетов;
- модуляция радиосигнала.

Функции шлюзов сети:
- демодуляция радиосигнала;
- обнаружение пакетов, передаваемых устройствами (поиск преамбулы);
- помехоустойчивое декодирование принятых пакетов;
- пересылка успешно принятых пакетов на сервер сети.

5.3 Функции канального уровня

Канальный уровень отвечает за механизм управления доступом устройств к радиоканалу (Media Access Control, MAC) и механизм управления логической связью (Logical Link Control, LLC) между устройствами и сервером сети.

Канальный уровень определяет:
- формат пакетов канального уровня;
- схему адресации конечных устройств;
- механизм множественного доступа конечных устройств к радиоканалу;
- механизм повторных передач пакетов;
- процедуру активации (аутентификации и регистрации) устройств на сервере;
- механизм криптографической защиты передаваемых данных (шифрование, контроль целостности и подлинности).

Канальный уровень протокола реализуется на конечных устройствах и сервере сети.

Функции конечных устройств:
- зашифрование и имитозащита данных;
- формирование и отправка пакетов активации;
- формирование и отправка пакетов данных.

Функции сервера сети:
- обработка пакетов активации (аутентификация и регистрация устройств на сервере);
- отбрасывание дубликатов пакетов, принятых от разных шлюзов;
- проверка целостности и подлинности пакетов;
- расшифрование пакетов;
- пересылка успешно расшифрованных пакетов на сервер приложений.

5.4 Функции уровня приложений

Сервер приложений отвечает за хранение и обработку прикладных данных, полученных от устройств.

В настоящем стандарте описывается физический и канальный уровень протокола OpenUNB. Описание уровня приложений выходит за рамки настоящего стандарта.

5.5 Требования к каналам связи

Беспроводной канал связи «устройство — шлюз» обеспечивает одностороннюю передачу информации от множества конечных устройств до множества шлюзов. Данный канал полностью определяется требованиями настоящего стандарта.

Канал связи «шлюз — сервер сети» обеспечивает двусторонний обмен информацией между шлюзом и сервером сети. Реализация данного канала выходит за рамки настоящего стандарта.

Канал связи «сервер сети — сервер приложений» обеспечивает двусторонний обмен информацией между сервером сети и сервером приложений. Реализация данного канала выходит за рамки настоящего стандарта.

При использовании настоящего стандарта в ситуациях, подпадающих под действие нормативных документов, регулирующих применение средств криптографической защиты информации на территории Российской Федерации, для обеспечения защиты каналов связи «шлюз — сервер сети», «сервер сети — сервер приложений» необходимо использовать один из криптографических протоколов, описываемых в рекомендациях по стандартизации.

6 Физический уровень

6.1 Формат пакетов

Формат пакетов физического уровня, передаваемых конечными устройствами, представлен на рисунке 2.

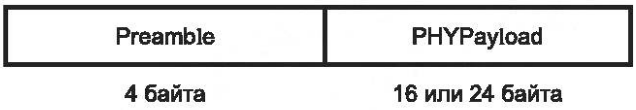

Рисунок 2 — Формат пакетов

Примечание — В описании формата пакетов порядок следования битов и байтов — от старшего к младшему (big endian).

Общая длина пакета физического уровня составляет 20 байт или 28 байт в зависимости от длины передаваемых полезных данных. Пакет включает два поля:

- Preamble;
- PHYPayload.

Поле Preamble содержит преамбулу пакета физического уровня. Алгоритм демодуляции, реализованный на шлюзе, использует данное поле для обнаружения пакета и синхронизации его последующей обработки. Поле Preamble имеет фиксированную длину 4 байта. В качестве преамбулы рекомендуется использовать различные m-последовательности, которые обладают выраженной автокорреляционной функцией. Рекомендуемое значение преамбулы физического уровня 0x97157A6F.

Поле PHYPayload содержит в себе сформированный пакет канального уровня, закодированный помехоустойчивым кодом. Длина поля PHYPayload с учетом избыточности помехоустойчивого кодирования (скорость кода равняется 1/2) составляет 16 байт или 24 байта.

6.2 Модуляция

Для передачи сформированного пакета физического уровня по радиоканалу должна использоваться одна из двух возможных схем модуляции сигнала:

- дифференциальная (относительная) двоичная фазовая манипуляция (DBPSK).

Примечание — Наилучшими помехоустойчивыми характеристиками обладает фазовая манипуляция (PSK). Однако использование дифференциального кодирования (DPSK) позволяет существенно упростить приемник за счет использования схемы некогерентного детектирования сигнала. При этом энергетический проигрыш (относительно PSK) составляет менее 0,5 дБ по уровню BER 10^{-6};

- двоичная частотная манипуляция (FSK).

Примечание — Этот тип модуляции рекомендуется использовать для подвижных устройств, так как он обладает лучшими характеристиками приема для «доплеровских» каналов.

6.3 Помехоустойчивое кодирование

На физическом уровне осуществляется помехоустойчивое кодирование пакетов с помощью полярного кода. Его использование позволяет шлюзам сети исправлять ошибки в принимаемых пакетах физического уровня, а также отбрасывать искаженные или ложные (шумовые) пакеты.

Примечание — Помехоустойчивое кодирование позволяет исправить только часть ошибок и не может детектировать все искаженные и ложные пакеты. Окончательное детектирование и отбрасывание искаженных и ложных (как шумовых, так и навязываемых злоумышленником) пакетов осуществляется на сервере сети в процессе расшифрования принятых пакетов.

Помехоустойчивое кодирование осуществляется для всего пакета канального уровня, который имеет размер 8 или 12 байт. Избыточность помехоустойчивого кода составляет 1/2, поэтому размер закодированного пакета составляет 16 или 24 байта.

Подробное описание используемого алгоритма помехоустойчивого кодирования представлено в приложении А.

7 Канальный уровень

7.1 Формат пакетов

Формат пакетов канального уровня, передаваемых конечными устройствами, представлен на рисунке 3.

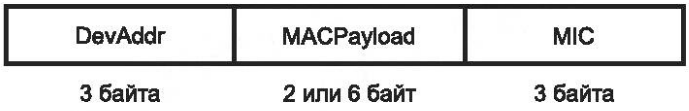

Рисунок 3 — Формат пакетов

Примечание — В описании формата пакетов порядок следования битов и байтов — от старшего к младшему (big endian).

Пакет данных канального уровня имеет длину 8 байт или 12 байт в зависимости от длины передаваемых полезных данных. Протокол предусматривает два типа пакетов канального уровня:

- служебный пакет активации, который передается конечным устройством при активации и содержит в себе служебную информацию, используемую для аутентификации конечного устройства и его регистрации на сервере сети;
- пакет полезных данных, в котором содержится информация уровня приложений, передаваемая конечным устройством на сервер приложений.

Пакет данных канального уровня включает три поля:

- DevAddr;
- MACPayload;
- MIC.

Поле DevAddr имеет фиксированную длину 3 байта. Данное поле содержит временный адрес устройства, значение которого зависит от типа пакета канального уровня:

- для служебных пакетов активации значение временного адреса устройства зависит только от идентификатора устройства DevID и вычисляется как DevAddr0 = CRC24 (DevID). Описание функции расчета циклического избыточного кода CRC24 представлено в приложении Б;
- для пакетов полезных данных значение временного адреса устройства пересчитывается каждую эпоху (раз в EPOCH_DURATION) с помощью функции криптографического преобразования, описание которой представлено в разделе 8.2.3.

Длина поля MACPayload составляет 2 байта или 6 байт.

Для служебных пакетов активации поле MACPayload содержит в себе двухбайтный номер текущей активации устройства N_a. В случае если длина поля MACPayload служебного пакета активации составляет 6 байт, номер N_a должен передаваться в двух младших байтах поля MACPayload, а старшие 4 байта поля MACPayload должны заполняться нулями.

Для пакетов полезных данных поле MACPayload содержит в себе информацию уровня приложений, передаваемую устройством на сервер. Эта информация зашифровывается на канальном уровне устройства и расшифровывается на сервере сети (см. раздел 8.2.4). Результат зашифрования поля MACPayload обозначается EncMACPayload. Внутренний формат этой информации выходит за рамки настоящего стандарта.

Поле имитовставки MIC (Message Integrity Code) имеет фиксированную длину 3 байта. Данное поле используется для проверки целостности и подлинности полей DevAddr и MACPayload. Значение поля MIC рассчитывается с помощью функции криптографического преобразования, описание которой представлено в разделе 8.2.5.

7.2 Адресация устройств

Для адресации конечных устройств в сети используются:

- DevID — постоянный идентификатор устройства, который хранится в памяти устройства и на сервере сети;
- DevAddr — временный (динамический) адрес устройства, который изменяется в процессе работы устройства и передается в пакетах канального уровня.

7.2.1 Идентификатор устройства DevID

Идентификатор устройства DevID записывается в энергонезависимую память устройства, а также хранится на сервере сети.

Идентификатор DevID должен иметь длину не менее 4 байт, а его формат может быть различным (например, в разных сетях может использоваться свой формат идентификаторов). Рекомендуется использовать идентификаторы DevID длиной не менее 128 бит.

7.2.2 Временный адрес устройства DevAddr

Временный адрес устройства DevAddr имеет фиксированную длину 3 байта.

Значение временного адреса устройства DevAddr изменяется в процессе работы устройства и передается в пакетах канального уровня:

- начальное значение временного адреса DevAddr0 = CRC24 (DevID). Этот адрес передается только в служебных пакетах активации, которые отправляются при активации устройства для его аутентификации и регистрации на сервере сети;
- для всех последующих пакетов, передающих полезные данные уровня приложений, устройство использует временный адрес DevAddr, который пересчитывается каждую эпоху (раз в EPOCH_DURATION) с помощью функции криптографического преобразования (см. раздел 8.2.3).

7.3 Выбор частоты передачи пакета

Для обеспечения множественного доступа устройств к радиоканалу в сети используется неслотированная частотная ALOHA. Частота передачи каждого пакета должна выбираться устройством случайно из всей полосы рабочих частот приема шлюзов. В случае если передача пакета осуществляется с повторами (см. раздел 7.4), для каждой очередной передачи должна выбираться случайная частота, отличная от частот, выбранных для предыдущих передач пакета.

Примечания

1 Особенностью настоящего стандарта является отсутствие сетки каналов — пакет данных принимается шлюзом вне зависимости от частоты, на которой была выполнена отправка. Это свойство позволяет использовать для формирования радиосигнала на конечных устройствах дешевые генераторы частоты, для которых невозможно обеспечить высокую точность синхронизации по времени и частоте.

2 Вследствие случайного выбора частоты передачи пакетов в сети возможно возникновение коллизий между пакетами различных устройств, передача которых пересекается по времени и частоте, что может служить причиной потерь пакетов.

7.4 Повторные передачи пакета

Для повышения надежности доведения информации до сервера сети допускается передавать каждый сформированный конечным устройством пакет несколько раз подряд. При этом каждая повторная передача пакета должна осуществляться на новой случайной частоте.

Передача служебных пакетов активации должна всегда осуществляться MAX_PKT_TX_NUM раз подряд.

Количество передач одного пакета полезных данных может быть от 1 до MAX_PKT_TX_NUM. Необходимость использования повторных передач для пакетов полезных данных определяется для каждого конечного устройства независимым образом.

8 Обеспечение информационной безопасности с использованием криптографических средств

8.1 Общие положения

Криптографическая защита реализуется на канальном уровне. Зашифрование данных осуществляется на конечном устройстве, а расшифрование — на сервере сети. На шлюзах сети вся информация обрабатывается только в зашифрованном виде.

Для каждого конечного устройства должны быть заданы:

- DevID — идентификатор устройства, определенный ранее в разделе 7.2.1;
- K_0 — долговременный ключ двоичной длины 256 бит. Значение ключа K_0 должно записываться в энергонезависимую память устройства, а также храниться на сервере сети.

В процессе работы каждое конечное устройство использует следующие переменные:

- счетчик N_a, содержащий число активаций устройства. Данный счетчик имеет длину 2 байта и может принимать значения от 0 до $2^{16} - 1$. Начальное значение счетчика $N_a = 0$ выставляется при записи в память устройства значений DevID и K_0;
- номер текущей эпохи N_e, который имеет длину 3 байта и принимает значения от 0 до $2^{24} - 1$;

- присваиваемый всем передаваемым пакетам номер N_n, который имеет длину 2 байта и принимает значения от 0 до $2^{16} – 1$.

Для того чтобы конечное устройство могло передавать на сервер пакеты полезных данных, оно должно предварительно пройти обязательную процедуру *активации* (регистрации на сервере сети). Описание процедуры активации устройства приведено в разделе 8.3.

От момента активации устройства время его работы разбивается на последовательные эпохи длительностью EPOCH_DURATION минут. В рамках каждой эпохи нумерация передаваемых пакетов производится независимо.

8.2 Базовые криптографические алгоритмы и ключевая система

8.2.1 Используемый блочный шифр

Для криптографической защиты информации должен использоваться алгоритм блочного шифрования с длиной блока 64 бит и длиной ключа 256 бит, регламентируемый ГОСТ Р 34.12.

8.2.2 Ключевая система

Долговременный ключ K_0 используется для формирования производных ключей: ключа активации и сессионных ключей, непосредственно используемых для шифрования и имитозащиты передаваемых пакетов.

Во время каждой активации устройства вырабатывается ключ активации K_a, определяемый равенством

$$K_a = CTR\ (K_0, N_a||0^{16}, 0^{256}).$$

Для каждой эпохи в рамках текущей активации вырабатываются сессионные ключи:

- ключ имитозащиты $K_m = CTR\ (K_a, 0x02||N_e, 0^{256})$;
- ключ шифрования $K_e = CTR\ (K_a, 0x03||N_e, 0^{256})$.

8.2.3 Формирование временного адреса DevAddr

Для каждой эпохи в рамках текущей активации вырабатывается временный адрес устройства:

$$DevAddr = MSB_{24}\ (ECB\ (K_a, 0x01||N_e||0^{32})).$$

8.2.4 Шифрование данных

Шифрование содержащихся в поле MACPayload пакета данных канального уровня осуществляется по формуле

$$EncMACPayload = CTR\ (K_e, N_n||0^{16}, MACPayload).$$

Расшифрование содержащихся в поле EncMACPayload пакета данных канального уровня осуществляется по формуле

$$MACPayload = CTR\ (K_e, N_n||0^{16}, EncMACPayload).$$

8.2.5 Имитозащита данных

Имитозащита данных, содержащихся в полях DevAddr и EncMACPayload, обеспечивается с использованием имитовставки. Имитовставка вырабатывается по формуле

$$MIC = CMAC_{24}(K_m, P).$$

Двоичный вектор P рассчитывается по формуле

$$P = DevAddr||EncMACPayload||N_n||0^{(len-16)}||len,$$

где len — однобайтная целочисленная константа, равная длине EncMACPayload в битах и принимающая значения 16 или 48. Двоичная длина вектора P равна 64 бита, если EncMACPayload содержит два байта, и равна 128 бит, если EncMACPayload содержит шесть байт.

8.3 Активация устройства

Для того чтобы конечное устройство могло передавать на сервер пакеты данных, оно должно предварительно пройти обязательную процедуру *активации* (регистрации на сервере сети).

Процедура активации конечного устройства состоит из следующих шагов:

- доставка идентификатора DevID конечного устройства и долговременного ключа K_0 на сервер сети,
- уведомление сервера сети о необходимости принятия пакетов данных от конечного устройства с заданным идентификатором,
- отправка конечным устройством сообщения активации о начале передачи прикладных данных.

Первые два шага приведенной последовательности действий решаются организационно-штатными или техническими мерами, описание которых выходит за рамки настоящего стандарта. Процедура отправки конечным устройством сообщения об активации полностью определяется требованиями настоящего стандарта.

Отправку сообщения об активации следует выполнять в следующих случаях:

- при первом подключении конечного устройства к источнику питания;
- при восстановлении конечного устройства после технических неполадок;
- при исчерпании счетчика эпох;
- при смене ключевой информации.

Алгоритм отправки сообщения об активации реализуется на канальном уровне.

Для отправки сообщения об активации устройство выполняет действия в следующей последовательности:

1) проверяется текущее значение счетчика активаций N_a. Если счетчик активаций уже достиг максимально возможного значения $2^{16} - 1$, то данное устройство больше не может использоваться без смены ключевой информации и выключается;

2) увеличивается значение счетчика активаций: $N_a = N_a + 1$;

3) вырабатывается ключ активации K_a;

4) вырабатывается сессионный ключ шифрования K_e для эпохи с номером $N_e = 0$;

5) вырабатывается сессионный ключ имитозащиты K_m для эпохи с номером $N_e = 0$;

6) формируется служебный пакет активации, в поле MACPayload которого содержится текущее значение счетчика активаций N_a (используется представление big-endian), а в качестве адреса DevAddr указывается значение DevAddr0 = CRC_{24}(DevID). Поле MACPayload передается в незашифрованном виде, а поле имитовставки MIC рассчитывается на ключе K_m для номера $N_n = 0$. Сформированный служебный пакет активации должен быть отправлен MAX_PKT_TX_NUM раз подряд для повышения вероятности его доведения;

7) устройство считает активацию успешно завершенной и может передавать пакеты полезных данных.

8.4 Отправка пакетов данных конечным устройством

Процедура формирования передаваемого пакета полезных данных должна состоять из следующей последовательности шагов:

1) определяется номер текущей минуты от момента активации устройства:

t_min = целое количество минут от (t – t_act),

где t — текущее время по часам устройства;

t_act — время активации устройства по часам устройства;

2) определяется номер текущей эпохи:

n_e = целая часть от деления t_min на EPOCH_DURATION;

3) если номер текущей эпохи n_e больше номера эпохи, в которой был передан предыдущий пакет данных, то:

- формируются новые сессионные ключи шифрования K_e и имитозащиты K_m;
- формируется новый временный адрес устройства DevAddr;

4) для пакета выбирается номер N_n, который должен удовлетворять следующим требованиям:

$$- cur\_min \le N_n \le cur\_min + MAX\_TX\_WINDOW - 1,$$

где cur_min — номер текущей минуты в рамках текущей эпохи:

cur_min = остаток от деления t_min на EPOCH_DURATION;

- номера пакетов N_n, переданных в рамках каждой отдельной эпохи, должны быть уникальными.

Если для пакета не может быть выбран номер N_n, удовлетворяющий определенным выше требованиям, то процедура формирования пакета завершается с ошибкой, и передача пакета блокируется;

5) поле MACPayload пакета зашифровывается;

6) для пакета вырабатывается имитовставка MIC;

7) процедура формирования пакета успешно завершается.

Приведенное выше правило определения номера пакета накладывает ограничение на интенсивность передачи пакетов устройством: количество пакетов, переданных за любые m минут подряд, не может превышать значение m + MAX_TX_WINDOW – 1.

При блокировке передаваемого пакета канальный уровень устройства должен сообщить об этом на уровень приложения. Алгоритм обработки такого события на уровне приложения может зависеть от специфики данных и настоящим стандартом не регламентируется.

Примечание — Точный алгоритм выбора номера N_n для передаваемого пакета настоящим стандартом не регламентируется. Пример реализации такого алгоритма представлен в приложении В.

8.5 Прием пакетов сервером сети

Сервер должен хранить список всех конечных устройств сети. Для каждого устройства в этом списке сервер должен хранить следующие переменные:

- dev_id — идентификатор устройства DevID, определенный в разделе 7.2.1;
- dev_addr_0 = CRC_{24}(DevID) — начальное значение временного адреса устройства, передаваемое в пакете активации;
- t_act — время активации устройства по часам сервера;
- n_a — текущий номер активации устройства;
- k_a — текущий ключ активации устройства;
- n_e — текущий номер эпохи на устройстве;
- dev_addr — временный адрес устройства для эпохи n_e;
- k_e — сессионный ключ шифрования для эпохи n_e;
- k_m — сессионный ключ имитозащиты для эпохи n_e;
- received_pkt_list — список номеров успешно принятых пакетов в эпохе n_e.

Примечания

1 В общем случае, протокол OpenUNB допускает возможность неверного определения сервером номера текущей эпохи n_e конечного устройства:

- вследствие рассинхронизации времени сервера и конечного устройства;
- вследствие задержки пакетов в канале «шлюз — сервер сети».

Для устранения данной проблемы рекомендуется хранить на сервере для каждого конечного устройства несколько возможных значений номеров эпохи n_e и соответствующих им значений временных адресов и ключей. Алгоритм работы сервера в соответствующем случае настоящим стандартом не регламентируется. Пример реализации такого алгоритма представлен в приложении В.

2 Допускается компенсация расхождения часов конечного устройства и сервера посредством дополнительной поправки для значения времени t_act, хранимой сервером для каждого конечного устройства. Алгоритм расчета значения дополнительной поправки настоящим стандартом не регламентируется. Пример реализации такого алгоритма представлен в приложении В.

Процедура обработки принятых пакетов на сервере должна состоять из следующей последовательности шагов:

1) на вход процедуры подаются следующие значения:

- pkt_dev_addr — адрес пакета,
- pkt_mac_payload — данные пакета,
- pkt_mic — код целостности пакета,
- t — время приема пакета;

2) осуществляется поиск устройств, для которых временный адрес dev_addr_0 или dev_addr совпадает с адресом pkt_dev_addr;

3) если на шаге 2 данной процедуры найдено хотя бы одно устройство, для которого dev_addr_0 совпадает с pkt_dev_addr, то принятый пакет обрабатывается как пакет активации. Для каждого найденного устройства выполняется процедура:

- если значение pkt_mac_payload ≤ n_a или pkt_mac_payload ≥ 216, то осуществляется переход к следующему устройству;
- вычисляются временные значения ключей активации tmp_k_a, шифрования tmp_k_e и имитозащиты tmp_k_m, соответствующие номеру активации pkt_mac_payload и номеру эпохи 0;
- с использованием ключа tmp_k_m проверяется корректность значения имитовставки pkt_mic принятого пакета:

если имитовставка не сходится, то осуществляется переход к следующему устройству;

в противном случае устройство считается успешно активированным и сервер сохраняет время активации (t_act = t), номер активации (n_a = pkt_mac_payload), номер эпохи (n_e = 0) и значения ключей (k_a = tmp_k_a, k_e = tmp_k_e и k_m = tmp_k_m), а также вычисляет временный адрес устройства dev_addr;

4) если на шаге 2 данной процедуры найдено хотя бы одно устройство, для которого dev_addr совпадает с pkt_dev_addr, то принятый пакет обрабатывается как пакет данных. Для каждого найденного устройства выполняется процедура:

- определяется интервал возможных значений номера n принятого пакета:

$$\max(0, cur\_min - prev\_n) \le n \le \min(EPOCH\_DURATION + MAX\_TX\_WINDOW - 2, cur\_min + MAX\_TX\_WINDOW - 1 + next\_n,$$

где cur_min — номер минуты в рамках эпохи n_e, в которой пакет был передан устройством: cur_min = = целое количество минут от (t – t_act – n_e*EPOCH_DURATION),

а значения переменных prev_n и next_n должны удовлетворять следующим требованиям:

$2 \le prev\_n \le MAX\_PREV\_N$;

$2 \le next\_n \le MAX\_NEXT\_N$;

- с использованием ключа k_m проверяется корректность значения имитовставки pkt_mic принятого пакета. При этом проверка имитовставки осуществляется для всего диапазона номеров n, определенного на предыдущем шаге, за исключением номеров, которые содержатся в списке received_pkt_list.

Если для какого-то номера n имитовставка сходится, то пакет расшифровывается на ключе k_e, а номер n добавляется в список received_pkt_list;

иначе осуществляется переход к следующему найденному устройству.

Примечание — Точный алгоритм определения сервером значений переменных prev_n и next_n настоящим стандартом не регламентируется, однако его реализация должна учитывать возможное расхождение времени на конечном устройстве и сервере сети. В приложении В представлен один из возможных примеров реализации такого алгоритма.

8.6 Таблица параметров

Параметры протокола OpenUNB приведены в таблице 1.

Таблица 1

Параметр	Описание	Рекомендованное значение по умолчанию
EPOCH_DURATION	Длительность одной эпохи в минутах	240 минут
MAX_PKT_TX_NUM	Максимальное количество передач одного пакета, которое может делать устройство для повышения надежности доведения пакета	6
MAX_TX_WINDOW	Размер окна выбора номера передаваемого устройством пакета	2
MAX_PREV_N	Максимально допустимый диапазон перебора номеров пакетов в прошлое	7
MAX_NEXT_N	Максимально допустимый диапазон перебора номеров пакетов в будущее	7

Приложение А
(обязательное)

Помехоустойчивое кодирование

А.1 Общее описание

На физическом уровне протокола осуществляется помехоустойчивое кодирование пакетов с помощью полярного кода.

Помехоустойчивое кодирование осуществляется для всего пакета канального уровня, который включает в себя поля DevAddr, MACPayload и MIC. Таким образом размер посылки для помехоустойчивого кода может равняться K = 64 или 96 бит (в зависимости от размера поля MACPayload).

Избыточность помехоустойчивого кода составляет 1/2, поэтому размер закодированного пакета может равняться N = 128 или 192 бита.

Кодирование передаваемых пакетов осуществляется на конечном устройстве. Описание процедуры кодирования представлено в разделе А.2.

Декодирование принимаемых пакетов осуществляется на шлюзе. Описание процедуры кодирования представлено в разделе А.3.

А.2 Кодирование

Перед непосредственным применением полярного кода для кодирования передаваемого пакета к нему необходимо применить циклический избыточный код длины 10 бит, задаваемый в виде 0x327. Данная операция осуществляется путем деления многочлена, коэффициенты которого соответствуют двоичному представлению передаваемого пакета, на многочлен 0x327. Двоичные коэффициенты, полученные в остатке данного деления, приписываются справа к передаваемому пакету (младшие 10 бит). Полученная таким образом последовательность (длиной 74 бита для K = 64 или 106 бит для K = 96) подается на вход кодера полярного кода и рассматривается как информационная последовательность для него.

Процедуру кодирования полярного кода выполняют в следующей последовательности:

1) Выбираются две строки u и cwd длины $\check{N} = 128$ для K = 64 и $\check{N} = 256$ для K = 96;

2) Позиции строки cwd, обозначенные в конфигурационном файле для данной длины символом 1, последовательно заменяют символами информационной последовательности. В случае K = 96 к символам информационной последовательности дополнительно добавляются 64 нуля справа (младшие биты) и для полученных 170 символов осуществляется аналогичная процедура;

3) Позиции строки u, обозначенные в конфигурационном файле для данной длины символом 0, принимаются равными 0;

4) Оставшиеся неизвестными позиции строк u и cwd определяются из условия $cwd = G_2^{\otimes \log_2 \check{N}} u$, где матрица $G_2 = \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}$, а оператор $\otimes \log_2 \check{N}$ соответствует произведению Кронекера матицы G_2 самой на себя $\log_2 \check{N}$ раз. Полученная в результате строка cwd является кодовым вектором, передаваемым в канал. В случае K = 96 64 символа, соответствующие последним символам 1 в конфигурационном файле, удаляются, после чего полученный кодовый вектор имеет требуемую длину N = 192 (данная процедура называется процедурой укорочения).

Примечание — Ввиду блочной структуры матрицы $G_2^{\otimes \log_2 \check{N}}$ возможно осуществление программной оптимизации вышеуказанной процедуры. В частности, возможно рекурсивное разделение строки u на две равные половины и сведение исходной системы к системам меньших размерностей. Данная операция может быть представлена как $uG_2^{\otimes n} = [u_1\ u_2] G_2^{\otimes n} = \left[(u_1 \oplus u_2) G_2^{\otimes n-1}\ u_2 G_2^{\otimes n-1} \right]$, где $\oplus$ соответствует сложению по модулю 2, а $G_2^{\otimes n-1}$ Кронекеровскому произведению матрицы G_2 самой с собой $n - 1$ раз.

А.3 Декодирование

Для декодирования рекомендуется использовать списочный алгоритм декодирования Тала-Варди на основе логарифмов отношения правдоподобия, описанный в [2]. При этом предполагается, что на вход декодера поступает строка, содержащая логарифмы отношений правдоподобия для каждого из символов принятой последовательности. Также входом декодера является длина кода N, конфигурационный файл для нее и максимальная длина используемого списка L.

Примечание — Значение L должно равняться степени двойки. Рекомендуемое значение L = 16, что обосновано балансом между вычислительной сложностью и корректирующей способностью кода. Если позволяют вычислительные ресурсы, то может быть использовано большее значение L (например, 32 или 64).

Перед непосредственным применением алгоритма Тала-Варди в случае N = 192 последовательно дополняем полученную на входе декодера строку с логарифмами отношений правдоподобия до длины $\check{N} = 256$ таким образом, что на позициях, соответствующих последним 64 символам 1 в конфигурационном файле, стоит величина 10 000.

По завершении работы алгоритма Тала-Варди умножаем слева все слова кодового списка на матрицу $G_2^{\otimes \log_2 \check{N}}$. Отметим, что по аналогии с шагом 4) раздела А.2 данная процедура может быть оптимизирована. Для всех полученных таким образом векторов осуществляем проверку циклического избыточного кода, а именно записываем биты, соответствующие первым K символам 1 в конфигурационном файле, в виде коэффициентов многочлена. В дальнейшем осуществляют деление данного многочлена на многочлен 0x327. В случае если остаток данного деления не совпадает со следующими 10 символами кодового слова, которые соответствуют символам 1 в конфигурационном файле, исключают данный кодовый вектор из списка.

Из всех оставшихся в списке кодовых векторов в качестве передаваемого выбирают тот, в котором метрика пути в алгоритме Тала-Варди минимальна.

В качестве передаваемой информации выбираем первые K бит, соответствующие символу 1 в конфигурационном файле.

А.4 Конфигурационные файлы

Конфигурационные файлы для полярного кода приведены в таблице А.1.

Таблица А.1

Тип модуляции	K	Конфигурационный файл
DBPSK	64	0x117037F01171FFF0017177F177FFFFF
	96	0x1011F013F7FFF011717FF17FFFFFF0001077F177F7FFF177FFFFFFFFFFFF
FSK	64	0x1701171FFF011F7FFF7FFFFFFF
	96	0x10003177F0017177F1FFFFFFF01171FFF7FFFFFFF7FFFFFFFFFFFFFFF

Примечания

1 Отметим, что здесь и далее перевод двоичных векторов в шестнадцатеричные осуществляется таким образом, что первый элемент двоичного вектора является наиболее значимым. Например, вектор 00111101 переводится в 0x3D.

2 Так как 10 бит циклического избыточного кода считаются информационными для используемого полярного кода, то число единиц в соответствующих конфигурационных файлах больше на 10. Кроме того, в случае K = 96 производится укорочение полярного кода длины $\check{N}$ = 256 на 64 символа, из-за чего конфигурационный файл для данной длины содержит 170 единиц и 86 нулей.

А.5 Тестовые последовательности

Тестовые последовательности приведены в таблице А.2.

Таблица А.2

Тип модуляции	K	Информационный вектор	Кодовый вектор
DBPSK	64	0xB3B4F7D43463B157	0x9FC611ED560FD7D4B383A43175455ECB
		0xC544F69D0AB8B8B8	0xE5F8E6512607169D53A0FA5C2DE2E278
	96	0xA1DA01890711D5361F6F8409	0xDC61144E18C5241A8DA30FB7A71F1A0BB000000000000000
		0x85825A732E2AF4DF91C977C8	0xF959F3615938552A4D9FC8E77CC7D787A000000000000000
FSK	64	0x50ED00C48388EA9B	0xC842978DCA617B40842C241C23AA6D74
		0xFB7C204C2C12D39	0xDA072188297F2DF0BB00261684B4E6A2
	96	0xA144551DF49ADE37F01F2E72	0xB452639D8861A051D909E5A357D26B78CB9BDF0179739216
		0x4AC0AB35BE3A20FF7A7D7FCA	0xA411DC18510AE530536272E636F8E883FB7FF7A76BFE54EA

Приложение Б
(обязательное)

Функция вычисления CRC24

CRC24 — циклический избыточный код длиной 24 бита, использующий полином 0x5D6DCB, начальное заполнение 0xFFFFFF и XOR выхода: 0xFFFFFF.

Далее представлен пример исходного кода, реализующего функцию вычисления CRC24:

```
#define CRC24_INIT 0xFFFFFFUL
#define CRC24_POLY 0x5D6DCBUL
#define CRC24_XOR_OUT 0xFFFFFFUL

uint32_t crc24(unsigned char* octets, size_t len)
{
   uint32_t crc = CRC24_INIT;
   int i;
   while (len--) {
      crc ^= (*octets++) << 16;
      for (i = 0; i < 8; i++) {
         crc <<= 1;
         if (crc & 0x1000000)
            crc ^= CRC24_POLY;
      }
   }
   return (crc & 0xFFFFFFL) ^ CRC24_XOR_OUT;
}
```

Контрольные примеры расчета CRC24 представлены в таблице Б.1.

Таблица Б.1

Данные для расчета	Значение CRC24
0x01020304	0xeb0466
0x04030201	0xfada5c
0x0a0b0c0d01020304	0x609b96
0x0a0b0c0d010203040000ff52000101fa	0xb02671

Приложение В
(справочное)

Примеры алгоритмов отправки и приема пакетов

В данном приложении содержатся примеры алгоритмической реализации процедур формирования и отправки пакетов данных конечными устройствами и приема пакетов данных сервером сети.

В.1 Отправка пакетов данных конечным устройством

Конечное устройство хранит следующие переменные:

- last_n_n — номер N_n последнего переданного конечным устройством пакета;
- last_n_e — номер эпохи N_e последнего переданного конечным устройством пакета.

Процедура формирования передаваемого пакета полезных данных должна состоять из следующей последовательности шагов:

1) определяется номер текущей минуты от момента активации устройства:

t_min = целое количество минут от (t – t_act),

где t — текущее время по часам устройства,

t_act — время активации устройства по часам устройства;

2) определяется номер текущей эпохи

n_e = целая часть от деления t_min на EPOCH_DURATION;

3) если n_e > last_n_e, то:
- формируются новые сессионные ключи шифрования K_e и имитозащиты K_m;
- формируется новый временный адрес устройства DevAddr;

4) для пакета определяется номер n_n с помощью следующей последовательности шагов:
- определяется номер текущей минуты в рамках текущей эпохи

cur_min = остаток от деления t_min на EPOCH_DURATION;

- если n_e > last_n_e, то n_n = cur_min;
- иначе если n_e == last_n_e, то:
  если last_n_n < cur_min, то n_n = cur_min;
  иначе если last_n_n < cur_min + MAX_TX_WINDOW — 1, то n_n = last_n_n + 1;
  иначе процедура формирования пакета завершается с ошибкой, и передача пакета блокируется.

5) last_n_n = n_n;
6) last_n_e = n_e;
7) поле MACPayload пакета зашифровывается;
8) для пакета вырабатывается имитовставка MIC;
9) алгоритм формирования пакета успешно завершается.

В.2 Прием пакетов данных сервером сети

В.2.1 Информация об активированных устройствах, хранимая на сервере сети

Сервер должен хранить список временных адресов DevAddr всех успешно активированных конечных устройств в сети.

Так как каждое конечное устройство вырабатывает новый временный адрес DevAddr для каждой эпохи, сервер также должен периодически обновлять адреса DevAddr всех активированных конечных устройств. Кроме того, так как между часами сервера и конечного устройства может быть ошибка синхронизации, то для каждого конечного устройства на границе двух его эпох сервер должен хранить сразу два адреса DevAddr (dev_addr_1 и dev_addr_2). Например:

- если по часам сервера для некоторого конечного устройства заканчивается текущая эпоха, то из-за ошибки синхронизации часов к этому моменту времени на конечном стройстве уже могла начаться следующая эпоха. Поэтому для такого конечного устройства сервер должен хранить адрес DevAddr как для текущей, так и для следующей эпохи;

- если по часам сервера для некоторого конечного устройства сейчас начало текущей эпохи, то из-за ошибки синхронизации часов в этот момент времени на конечном устройстве еще может быть предыдущая эпоха. Поэтому для такого конечного устройства сервер должен хранить в списке адреса DevAddr как для текущей, так и для предыдущей эпохи.

Так как используемая схема вычисления временных адресов DevAddr не гарантирует их уникальность, на сервере допустимо появление одинаковых адресов DevAddr для различных конечных устройств. Разрешение возможных коллизий временных адресов разных конечных устройств осуществляется на сервере сети при расшифровании принятых пакетов на этапе проверки имитовставки MIC (см. раздел В.2.3).

Представленный в настоящем приложении алгоритм обработки пакетов данных, принятых сервером сети, предполагает, что сервер хранит для каждого успешно активированного конечного устройства следующую информацию:

- dev_id — идентификатор устройства DevID;
- dev_addr_0 = CRC_{24}(DevID) — начальное значение временного адреса устройства, передаваемое в пакете активации;
- t_act — время активации конечного устройства по часам сервера;
- d_t — поправка времени, компенсирующая расхождение часов сервера и конечного устройства;
- n_a – текущий номер активации устройства;
- k_a – текущий ключ активации устройства;
- last_pkt_rx_time — время приема последнего пакета от устройства;
- n_e_1 — первый номер эпохи, в рамках которой сервер ожидает получения пакетов от данного конечного устройства;
- n_e_2 — второй номер эпохи, в рамках которой сервер ожидает получения пакетов от данного конечного устройства. Значение n_e_2 = n_e_1 + 1;
- dev_addr_1 — временный адрес устройства для эпохи n_e_1;
- dev_addr_2 — временный адрес устройства для эпохи n_e_2;
- k_e_1 — сессионный ключ шифрования для эпохи n_e_1;
- k_e_2 — сессионный ключ шифрования для эпохи n_e_2;
- k_m_1 — сессионный ключ имитозащиты для эпохи n_e_1;
- k_m_2 — сессионный ключ имитозащиты для эпохи n_e_2;
- received_pkt_list_1 — список номеров успешно принятых пакетов в эпохе n_e_1;
- received_pkt_list_2 — список номеров успешно принятых пакетов в эпохе n_e_2.

В.2.2 Алгоритм пересчета адресов и ключей для активированных устройств

Далее представлен пример алгоритма обновления временных адресов и сессионных ключей для активированных конечных устройств на сервере.

Один раз за время, равное EPOCH_DURATION/2, сервер для каждого успешно активированного конечного устройства выполняет следующие действия (здесь t — текущее время по часам сервера):

- если номер текущей эпохи на данном конечном устройстве равен n_e_2 (т. е. целая_часть((t – t_act + d_t)/ /EPOCH_DURATION) == n_e_2), и прошло уже более 1/4 длительности текущей эпохи (дробная_часть((t – t_act + + d_t)/EPOCH_DURATION) > 0,25), то:
- n_e_1 = n_e_2;
- n_e_2 = n_e_2 + 1;
- dev_addr_1 = dev_addr_2;
- dev_addr_2 равняется новому временному адресу DevAddr, вычисленному для эпохи с номером n_e_2;
- k_e_1 = k_e_2;
- k_e_2 равняется новому сессионному ключу шифрования, вычисленному для эпохи с номером n_e_2;
- k_m_1 = k_m_2;
- k_m_2 = равняется новому сессионному ключу имитозащиты, вычисленному для эпохи с номером n_e_2;
- received_pkt_list_1 = received_pkt_list_2;
- received_pkt_list_2 равняется пустому списку.

Примечание — В случае использования данного алгоритма для каждого конечного устройства сервер всегда хранит два временных адреса DevAddr. При этом в интервале времени от 1/4 от начала текущей эпохи до 3/4 от начала текущей эпохи сервер удаляет всю информацию о предыдущей эпохе и добавляет информацию о следующей эпохе. Данный алгоритм может быть оптимизирован. Например, можно сделать так, чтобы сервер хранил два временных адреса для конечного устройства только в пределах некоторого малого интервала на границе смены эпох, а в остальное время хранил только один временный адрес. Это уменьшит общее количество временных адресов конечных устройств, хранимое на сервере, и ускорит поиск конечного устройства-отправителя для принятого пакета. Однако в таком случае на сервере вырастет вычислительная нагрузка, вызванная необходимостью более частого поиска конечных устройств, для которых требуется обновить хранимые временные адреса.

В.2.3 Алгоритм обработки принятого пакета данных

При получении пакета от шлюза сервер должен проверить, является ли этот пакет пакетом данных, переданным одним из успешно активированных устройств, и если является, то расшифровать его. Для этого сервер выполняет следующую последовательность шагов:

1) на вход процедуры подаются следующие значения:
- pkt_dev_addr — адрес пакета,
- pkt_mac_payload — данные пакета,
- pkt_mic — код целостности пакета,
- t — время приема пакета по часам шлюза,

2) в списке активированных конечных устройств осуществляется поиск устройств, для которых временный адрес (dev_addr_1 или dev_addr_2) совпадает с адресом принятого пакета pkt_dev_addr;

3) если на шаге 2 алгоритма устройств не найдено, то алгоритм завершается;

4) для каждого конечного устройства, найденного на шаге 2 алгоритма, выполняется процедура:

- в зависимости от того, какой из временных адресов устройства (dev_addr_1 или dev_addr_2) совпал с pkt_dev_addr на шаге 2 алгоритма, для устройства определяется номер эпохи, сессионные ключи и список успешно принятых пакетов:

  - если pkt_dev_addr == dev_addr_1, то:
    n_e = n_e_1;
    k_e = k_e_1;
    k_m = k_m_1;
    received_pkt_list = received_pkt_list_1;
  - если pkt_dev_addr == dev_addr_2, то:
    n_e = n_e_2;
    k_e = k_e_2;
    k_m = k_m_2;
    received_pkt_list = received_pkt_list_2;

- определяется интервал возможных значений номера n принятого пакета:

$$\max(0, cur\_min - prev\_n) \le n \le \min(EPOCH\_DURATION + MAX\_TX\_WINDOW - 2, cur\_min + MAX\_TX\_WINDOW - 1 + next\_n,$$

где:

- cur_min — номер минуты в рамках эпохи n_e, в которой пакет был передан устройством: cur_min = целое количество минут от (t – t_act – n_e*EPOCH_DURATION),
- prev_n = next_n = 2 + rx_window,
- rx_window — размер окна поиска номера пакета, который определяется сервером исходя из возможной рассинхронизации часов сервера и устройства (см. раздел В.2.4):

rx_window = целая часть от деления (t — last_pkt_rx_time) на RX_WINDOW_UPDATE_PERIOD.

При этом если значения prev_n и next_n превышают предельно допустимые значения MAX_PREV_N и MAX_NEXT_N, то работа данного конечного устройства должна быть заблокирована на сервере (оно считается неисправным, и должна быть проведена либо его повторная активация, либо замена) и осуществляется переход к следующему найденному устройству;

- с использованием ключа k_m проверяется корректность значения имитовставки pkt_mic принятого пакета. При этом проверка имитовставки осуществляется для номеров n из диапазона, определенного на предыдущем шаге, за исключением номеров, которые содержатся в списке received_pkt_list;

5) если на шаге 4 алгоритма имитовставка совпала сразу для нескольких устройств или не совпала ни для одного из устройств, то принятый пакет должен быть отброшен, и алгоритм завершается;

6) иначе если на шаге 4 алгоритма для одного из устройств совпала имитовставка для некоторого номера n, то обрабатываемый пакет считается принятым от этого устройства, и выполняются следующие действия:

- пакет расшифровывается на ключе k_e,
- номер n добавляется в список received_pkt_list,
- обновляется значение last_pkt_rx_time = t,
- при необходимости обновляется значение d_t:
  если номер n < cur_min – 1, то d_t = d_t – (cur_min – 1 – n);
  иначе если номер n > cur_min + MAX_TX_WINDOW, то d_t = d_t + (n – cur_min + MAX_TX_WINDOW);
- алгоритм обработки пакета данных завершается.

В.2.4 Синхронизация часов сервера на часы конечного устройства

Алгоритм обработки принятых пакетов данных на сервере сети (см. раздел В.2.3) основан на синхронизации часов сервера и конечного устройства: зная время активации конечного устройства t_act, сервер может для любого момента времени t определить текущее время на конечном устройстве, из которого рассчитать номер текущей эпохи на конечном устройстве N_e, его временный адрес DevAddr, ключи K_e и K_m и номер N_n переданного конечным устройством пакета.

Однако синхронизация часов сервера и конечного устройства осуществляется с ошибкой, значение которой включает в себя следующие составляющие:

- ошибка синхронизации часов сервера и шлюзов. Время приема всех пакетов определяется на шлюзах сети. Часы шлюзов и сервера должны быть засинхронизированы на абсолютное время, например по протоколу NTP или по сигналу от ГНСС (GPS/ГЛОНАСС). Предполагается, что ошибка синхронизации часов шлюзов и сервера не превышает одной секунды и поэтому ею можно пренебречь;
- ошибка начальной синхронизации сервера и устройства. Начальная синхронизация сервера с конечным устройством осуществляется по времени получения от него служебного пакета активации. Однако так как передача служебного пакета активации осуществляется MAX_PKT_TX_NUM раз подряд, а сервер не имеет возможности

определить, какой именно из этих пакетов был принят, то ошибка начальной синхронизации времени между сервером и конечным устройством не превышает 15 секунд (каждый пакет может быть передан не более MAX_PKT_TX_NUM = 6 раз подряд, а передача одного пакета длится 1,6 или 2,24 секунды для пакета длиной 20 и 28 байт соответственно);

- накопленный уход часов конечного устройства от момента его текущей активации, вызванный нестабильностью используемого опорного генератора на устройстве. Все конечные устройства должны иметь стабильность часов не хуже 170 ppm. Таким образом, допускается, что уход часов конечного устройства, например, за каждые сутки может достигать почти 15 секунд, а за каждый год — почти 1,5 часа. Для компенсации накопленного ухода часов конечных устройств сервер должен для каждого конечного устройства определять значение корректирующей поправки d_t в минутах. Пересчет значения d_t для конечного устройства может осуществляться только в случае успешного приема пакетов от этого конечного устройства (см. описание далее). Значение d_t используется сервером для определения времени по часам конечного устройства. Пусть текущее время на сервере t, тогда текущее время на конечном устройстве вычисляется как t – t_act + d_t. Например, если d_t = 5, то это означает, что за время работы конечного устройства (прошедшего от момента его текущей активации) его часы ушли вперед на 5 минут, а если d_t = –10, то это означает, что за время работы конечного устройства (прошедшего от момента его текущей активации) его часы отстали на 10 минут.

Из-за ошибки синхронизации часов сервер не может точно определить номер минуты cur_min, в которую принимаемые пакеты были отправлены конечным устройством, а значит, не может определить и их номер N_n. Поэтому при приеме пакетов сервер должен проверять сразу несколько возможных значений N_n из диапазона от cur_min – – prev_n до cur_min + MAX_TX_WINDOW – 1 + next_n (см. раздел В.2.3). Значения prev_n и next_n определяются исходя из следующих правил:

- начальные значения prev_n = 2 и next_n = 2. Это обусловлено тем, что сразу после активации устройства ошибка определения cur_min для принимаемых от него пакетов не превышает одной минуты. А так как на устройстве значение N_n может выбираться из диапазона от cur_min до cur_min + MAX_TX_WINDOW – 1, то для приема первых пакетов серверу достаточно проверять имитовставку только для номеров от cur_min – 1 до cur_min + + MAX_TX_WINDOW. Однако со временем часы конечного устройства могут уйти из-за нестабильности опорного генератора. Поэтому в случае, если имитовставка принимаемых пакетов не сходится для номеров от cur_min – 1 до cur_min + MAX_TX_WINDOW, сервер должен дополнительно проверить номера cur_min – 2 и cur_min + MAX_TX_WINDOW + 1. Если для очередного пакета имитовставка сойдется для номера cur_min – 2, то сервер делает вывод о том, что с момента последнего пересчета поправки d_t часы конечного устройства накопили отставание на 1 минуту, и поэтому d_t уменьшается на 1. Аналогично, если для очередного пакета имитовставка сойдется для номера cur_min + MAX_TX_WINDOW + 1, то сервер делает вывод о том, что с момента последнего пересчета поправки d_t часы конечного устройства убежали вперед на 1 минуту, и поэтому d_t увеличивается на 1. Таким образом, представленный выше механизм позволяет серверу в случае стабильного приема пакетов от устройства постоянно синхронизировать свое время под часы устройства за счет пересчета поправки d_t;

- однако если в течение долгого времени сервер не получает от устройства пакетов, то их часы могут со временем разойтись: при предельно допустимой стабильности часов устройства 170 ppm, часы устройства могут уходить на 1 минуту вперед или назад за каждый RX_WINDOW_UPDATE_PERIOD = 4 суток. Поэтому в случае если на сервере не было ни одного успешно принятого пакета от какого-то конечного устройства в течение времени M * RX_WINDOW_UPDATE_PERIOD, то за это время часы этого устройства могли уйти на M минут, и поэтому значения max_prev_n и max_next_n должны быть дополнительно увеличены на M. Таким образом, в общем случае должны использоваться значения prev_n = next_n = 2 + rx_window, где rx_window = целая часть от деления (t — last_pkt_rx_time) на RX_WINDOW_UPDATE_PERIOD, где t — текущее время, а last_pkt_rx_time — время приема последнего пакета от соответствующего устройства (см. раздел В.2.3). Если значения prev_n и next_n превышают предельно допустимые значения MAX_PREV_N и MAX_NEXT_N, то работа данного конечного устройства должна быть заблокирована на сервере (оно считается неисправным, и должна быть проведена либо его повторная активация, либо замена).

Приложение Г
(справочное)

Контрольные примеры

В данном приложении содержатся контрольные примеры формирования служебных пакетов активации и пакетов полезных данных с использованием алгоритмов, представленных в настоящем стандарте.

Г.1 Служебные пакеты активации

Контрольные примеры формирования служебных пакетов активации приведены в таблице Г.1.

Таблица Г.1

Номер примера	Идентификатор конечного устройства DevID и ключ K_0	Номер активации N_a	Сформированный служебный пакет активации
1	DevID=0x67C6697351FF4AEC29CDBAABF2FBE346 K_0=0x7CC254F81BE8E78D765A2E63339FC99A 66320DB73158A35A255D051758E95ED4	0x3DAB	0x5427A53DAB78D645
2		0x3DAC	0x5427A53DACCA7E61
3	DevID=0xB2CDC69BB454110E827441213DDC8770 K_0=0xE93EA141E1FC673E017E97EADC6B968F38 5C2AECB03BFB32AF3C54EC18DB5C02	0x481A	0xE6CB3E481A789741
4		0x481B	0xE6CB3E481B6D3A4B

Г.2 Пакеты полезных данных

Контрольные примеры формирования пакетов полезных данных приведены в таблице Г.2.

Таблица Г.2

Номер примера	Идентификатор конечного устройства DevID, ключ K_0, номер активации N_a, номер эпохи N_e, номер пакета N_n	Полезные данные Payload	Сформированный пакет полезных данных
1	DevID=0xFBFAAA3AFB29D1E6053C7C9475D8BE61 K_0=0x89F95CBBA8990F95B1EBF1B305EFF700 E9A13AE5CA0BCBD0484764BD1F231EA8 N_a=0x3C5A N_e=0x9ABBB7 N_n=0x0001	0x1C7B	0x4C024F29372A189B
2		0x64C514735AC5	0x4C024F5189B222AFA2 59E8AB
3	DevID=0x79633B706424119E09DCAAD4ACF21B10 K_0=0xAF3B33CDE3504847155CBB6F2219BA9B7DF5 0BE11A1C7F23F829F8A41B13B5CA N_a=0x21FC N_e=0x322365 N_n=0x0001	0x4EE8	0xA79BD153DDAC7782
4		0x983238E0794D	0xA79BD18507466B0E84 7FB9BE

Библиография

[1] ETSI TR 103 435-2017 Системный справочный документ (SRdoc). Устройства малой дальности (SRD). Технические характеристики для крайней узкой полосы (UNB) SRDs, работающей в спектре UHF ниже 1 ГГц

[2] A. Balatsoukas-Stimming, M. B. Parizi and A. Burg, LLR-Based Successive Cancellation List Decoding of Polar Codes. In IEEE Transactions on Signal Processing, vol. 63, no. 19, pp. 5165—5179, Oct. 1, 2015, doi: 10.1109/TSP.2015.2439211

УДК 004.738;006.354

ОКС 35.020
35.110

Ключевые слова: информационные технологии, интернет вещей, протокол беспроводной передачи данных, сверхузкополосная модуляция радиосигнала

Редактор *Л.В. Коретникова*
Технический редактор *В.Н. Прусакова*
Корректор *О.В. Лазарева*
Компьютерная верстка *Е.О. Асташина*

Сдано в набор 15.03.2023. Подписано в печать 22.03.2023. Формат 60×84⅛. Гарнитура Ариал.
Усл. печ. л. 2,79. Уч.-изд. л. 2,12.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

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