ОСНОВЫ БЕЗОПАСНОСТИ СИСТЕМ IoT

ПЕТР ПАРХОМЕНКО, инженер

В статье кратко рассматриваются вопросы обеспечения безопасности на всех уровнях в системах интернета вещей (IoT). Особое внимание уделено защите ключей – необходимо не только обеспечить надежное место для их хранения, но и ограничить доступ к ним со стороны приложений. Кроме того, рассматриваются угрозы, возникающие при перезагрузке, включении или нормальной работе узла. Наконец, описываются риски, возникающие при работе с удаленными ресурсами и облачными сервисами.

АЛГОРИТМЫ КРИПТОГРАФИИ

С одной стороны, защита простых сетевых узлов, например датчиков температуры, может показаться избыточной. Однако без нее они становятся возможной точкой входа злоумышленников в корпоративную сеть.

Рассмотрим классы криптографических алгоритмов и их роль в обеспечении безопасности.

Криптографические алгоритмы можно разделить на три следующие категории.

- *Симметричные*. Используется один и тот же секретный код для шифрования текстового сообщения в кодовое. При приеме производится дешифрование. Симметричные шифраторы применяются для обеспечения конфиденциальности.
- *Асимметричные*. Используется парный набор приватных и публичных ключей для шифрования и восстановления сообщений. Как правило, алгоритмы этого типа входят в состав расширенных протоколов безопасности и применяются для согласования ключей или в качестве электронной подписи. Они обеспечивают конфиденциальность, аутентификацию и невозможность отказа.
- *Алгоритмы хэширования*. Исходное сообщение упаковывается в более компактное значение уникальной фиксированной длины, которое называется хэшем, профилем или подписью. Такое преобразование играет ключевую роль в верификации целостности сообщения, т. е. для проверки на отсутствие изменений. Оно применяется в большом количестве протоколов, основанных на кодах аутентификации сообщений (MAC) и сообщений с хэш-кодированием (HMAC) либо на функции формирования ключа (KDF) и т. д.

Алгоритмы криптографии на основе ключей генерируют кодированный текст, который невозможно либо экономически нецелесообразно дешифровать без ключа. При этом алгоритмы хэширования должны быть достаточно быстрыми, чтобы генерировать абсо-

Таблица. **Примеры криптопроцессоров и контроллеров**

Производитель	Модель	Тип	Алгоритм
Maxim Integrated	MAX32631	32-разр. МК	AES (симм.), DSA (асимм.)
	MAX32520	32-разр. МК	AES (симм.), SHA (хэш), ECDSA (асимм.)
Microchip Technology	PIC 24F XLP	16-разр. МК	AES (симм.), 3DES (симм.)
	PIC 32MZ	32-разр. МК	AES (симм.), 3DES (симм.), SHA (хэш), HMAC (хэш)
	SAM9X60	32-разр. МП	AES (симм.), 3DES (симм.), SHA (хэш), HMAC (хэш)
Texas Instruments	SimpleLink	МК	AES, ECDH, ECDSA, SHA

Рис. 1. **Установление сессии TLS 1.2 с использованием ряда протоколов аутентификации, обмена ключами и данными**

лютно разные хэши для сообщений, которые имеют хотя бы незначительные различия.

Криптографические функции встроены в процессоры или специализированные ИС (см. табл.), что позволяет разработчикам больше времени заниматься другими задачами. При этом обеспечивается дополнительная защита, поскольку данные находятся и обрабатываются внутри компонента. Таким образом, информацию сложнее «подслушать». Встроенные криптоускорители позволяют разгрузить основной процессор. Следует также отметить высокую надежность аппаратных средств защиты. Например, генераторы случайных чисел основаны на значениях шумового напряжения в полупроводниках. Такие параметры невозможно предугадать или вычислить.

ПРОТОКОЛ АУТЕНТИФИКАЦИИ

Надежные криптографические протоколы имеют фундаментальное значение для протоколов безопасности высокого уровня, используемых в приложении. Так, транспортный протокол безопасности (Transport Layer Security, TLS) обеспечивает конфиденциальность и аутентичность между IoT-клиентом и хост-сервером, осуществляя длинные процедуры обмена данных (см. рис. 1).

Аутентификация обеспечивается путем идентификации сервера по сертификатам безопасности, в которых содержатся публичные ключи, присвоенные каждому участнику. Участники отсылают сообщения, защищенные приватным ключом. Поскольку полученный публичный ключ позволяет расшифровать сообщение, защищенное приватным ключом, участники могут подтвердить, что предоставляющее сертификат устройство действительно является его обладателем.

На следующей стадии участники выполняют серию транзакций для создания общего ключа сессии, который защищает передаваемые сообщения. Таким образом обеспечивается конфиденциальность обмена данными в рамках сессии. Большое количество настроек позволяет варьировать уровень безопасности. Например, шифрование может применяться до установления сессии, как показано на рисунке 2.

Уровень защищенности данных характеризуется с помощью параметра под названием «сила защиты». Она определяется количеством x бит при допущении, что для незаконного получения приватного ключа потребуется примерно 2^x операций. Таким образом, разные алгоритмы могут обеспечивать один и тот же уровень безопасности при разной длине ключа.

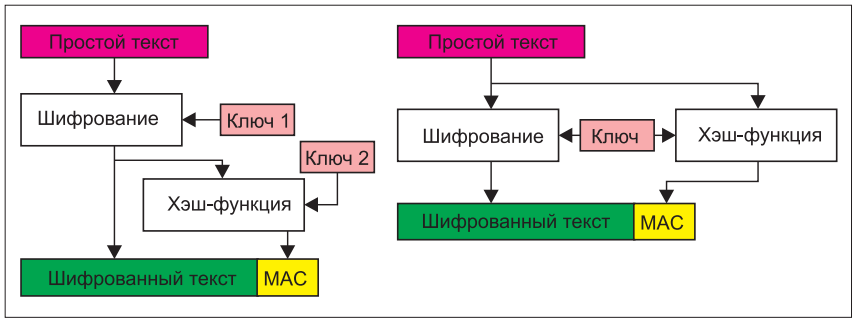

Рис. 2. **Примеры выполнения предварительного шифрования**

РОЛЬ СЕКРЕТНЫХ КЛЮЧЕЙ

Любой алгоритм защиты будет работать только тогда, когда ключи шифрования надежно сохранены. В теории это звучит достаточно просто, однако на практике защиту необходимо обеспечивать во всех режимах работы (ожидание, передача, обработка) и во всех узлах сети.

Как правило, хранение ключей и прочих секретных данных производится в энергонезависимой памяти. Главным ядром, обеспечивающим безопасность, может быть и специализированная ИС, и криптопроцессор.

Интегральные схемы представляют собой автономную подсистему и служат для разгрузки основного процессора. Связь между ЦП и ИС происходит по последовательной шине (см. рис. 3).

Программные методы защиты обеспечивают уровень абстракции, в результате чего по системе передаются не сами ключи, а их идентификаторы. Таким образом, сами секретные данные мало участвуют в операциях.

Не в любой системе допустимо использовать ИС безопасности из-за требований к размеру, стоимости, типу компонентов и т. д. Вместо них могут использоваться криптопроцессоры, содержащие средства защиты данных, и ускорители криптографических алгоритмов.

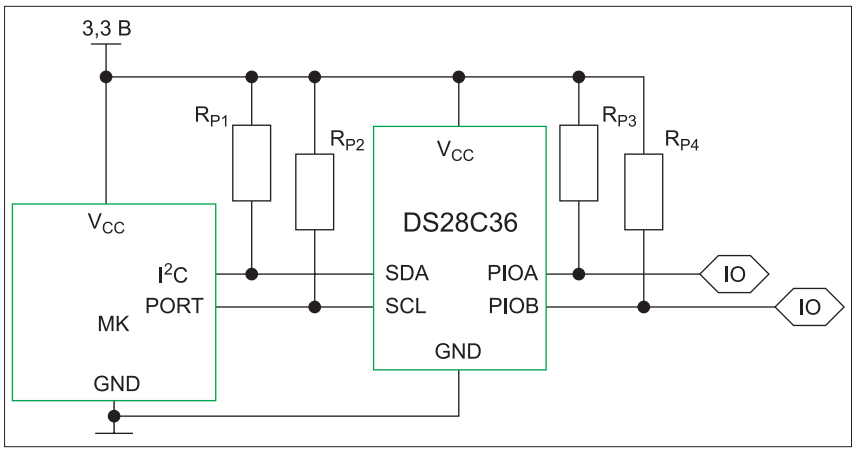

Рис. 3. **Подключение ИС безопасности Maxim DS28C36 к процессору по шине I2C. Защищенные порты ввода–вывода общего назначения служат для передачи сигнала об успешной или неудачной аутентификации в главном процессоре**

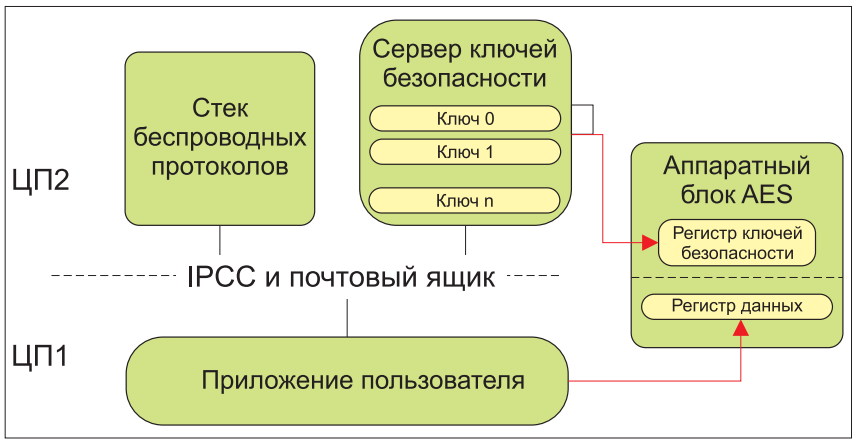

Рис. 4. **Двуядерный процессор STM32WB55 от STMicroelectronics позволяет получать напрямую ключи безопасности только микроконтроллеру (ЦП2). Проверенные приложения запускаются на ядре (ЦП1) и не имеют прямого доступа к ключам**

Например, двуядерный беспроводной модуль STM32WB55 от STMicroelectronics с процессором Arm Cortex-M4 выполняет функции хост-процессора, обмениваясь данными с микроконтроллером Arm Cortex-M0+. Ключи хранятся в защищенной памяти внутри микроконтроллера (см. рис. 4).

ФИЗИЧЕСКИ НЕКЛОНИРУЕМЫЕ ФУНКЦИИ

Несмотря на многоуровневое шифрование, сбои неизбежны. Мошенники используют утечки данных для быстрой продажи их на черном рынке. В результате злоумышленники получают относительно «легальный» доступ.

В настоящее время в криптопроцессорах и ИС безопасности все чаще используется технология физически неклонируемой функции (physical unclonable function, PUF). На текущий момент она, вероятно, является наиболее эффективной. Вместо прямой загрузки ключа защиты она подразумевает использование ключей, основанных на уникальных физических параметрах устройства. Существует немало практических реализаций такого подхода, однако все они основаны на одном и том же принципе.

Известно, что производственные отклонения характеристик присущи всем устройствам, и они воспроизводимы при условии правильной работы системы. Далее мы увидим, как их можно использовать в качестве ключей безопасности. Другими словами, каждое устройство с функцией PUF имеет встроенный секретный код, который нигде не хранится. При этом при попытке проникнуть в устройство с целью получения кода характеристики, используемые для его генерации, меняются, т. е. злоумышленник увидит другое значение.

Например, функция ChipDNA, используемая в микроконтроллере MAX32520 от Maxim Integrated и некоторых ИС безопасности, основана на массиве аналоговых PUF-элементов и логических схемах, которые генерируют ключ (см. рис. 5).

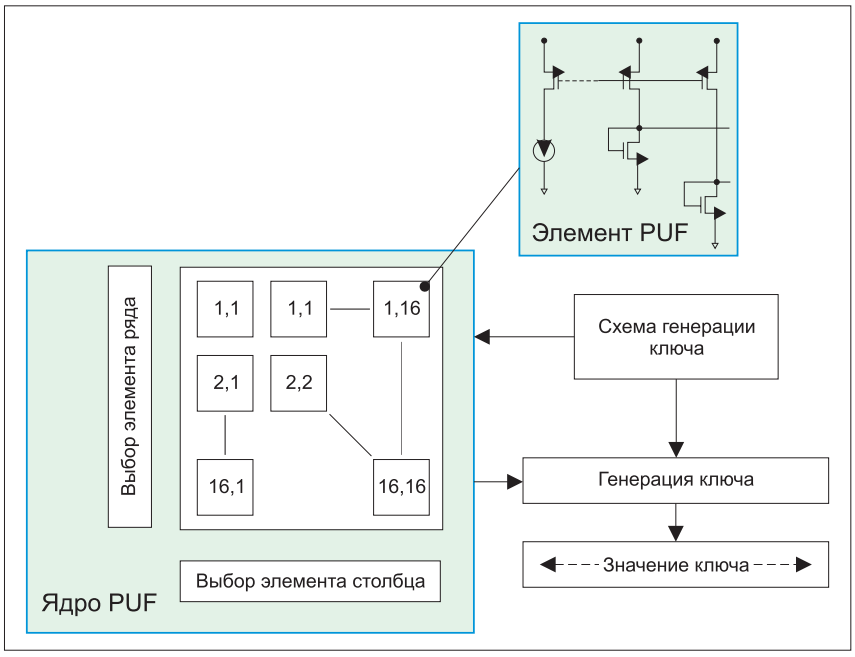

Рис. 5. **Схема генерации случайного ключа безопасности ChipDNA от Maxim Integrated. Значение ключа рассчитывается в управляющей логической схеме на основе случайных состояний массива элементов PUF**

Рис. 6. **Процессор с TrustZone STM32L552VET6 от STMicroelectronics обеспечивает комбинацию режимов, позволяющих изолировать поверенное ПО, например образы загрузки, от непроверенного кода приложений (сторонних стеков коммуникационных протоколов радиосвязи)**

БЕЗОПАСНОСТЬ ПЕРЕЗАГРУЗКИ

В типичной встраиваемой системе системный сброс, вызываемый падением напряжения питания или критическим исключением программного уровня, в конечном счете вызывает программный процесс перезагрузки для повторной загрузки образа прошивки из энергонезависимой памяти. В общем случае перезагрузка ПО является важным механизмом безопасности, позволяющим восстановить работоспособность после случайного или намеренного сбоя. Конечно, перезагрузка не решает проблему, если злоумышленник проник в аппаратную прошивку. Для предотвращения подобных вмешательств используется концепция, основанная на корне доверия (root of trust). Она гарантирует, что перезагрузка производится с безопасной прошивки.

Для выяснения того, взломана ли аппаратная прошивка, используется цифровая подпись программ. Производитель присваивает образу прошивки приватный ключ, который впоследствии хранится на предприятии. В комплекте с прошивкой поставляется публичный ключ. Во время перезагрузки он позволяет выявить, соответствует образ прошивки заводскому или он взломан. Следует иметь в виду, что публичный код должен быть надежно защищен. Зная этот код, злоумышленник может заменить его собственной парой приватного и публичного ключей, т. е. при проверке взломанная система будет определена как подлинная.

УГРОЗЫ ВО ВРЕМЯ РАБОТЫ

Во время работы системы может быть получено и запущено вредоносное ПО или код, который откроет доступ к системе. Злоумышленники используют слабые места одних узлов системы, чтобы произвести атаку на другие. Например, при переполнении буфера программные приложения позволяют записывать большие объемы входных данных в другие области памяти, и процессор впоследствии выполняет эти коды.

Количество уязвимых мест растет, поскольку функционал систем постоянно расширяется. Для защиты следует применять многоуровневые схемы

проверок. Рассмотрим их на примере архитектуры Arm TrustZone. Она обеспечивает безопасную загрузку, защищенность кода, данных и памяти.

Архитектура TrustZone подразумевает несколько режимов работы процессоров: безопасный, незащищенный, режим куратора и потоковый. Они различаются механизмами защиты.

В безопасном и незащищенном режимах производится разделение между процессами, которым можно доверять, и непроверенными процессами. Режим куратора и потоковый режим обеспечивают более глубокое разделение.

В режиме куратора ПО имеет привилегированный режим выполнения. Это необходимо для ОСРВ, а также для получения доступа к образу загрузки, ключам безопасности или другим критичным ресурсам. Наоборот, в потоковом режиме ПО имеет низкий приоритет исполнения. В этом режиме запускается код приложений.

При использовании комбинации приведенных режимов обеспечивается упоминавшаяся многоуровневая защита. Например, процессор STM32L552VET6 от STMicroelectronics позволяет отделить безопасный код от потенциально вредоносного (см. рис. 6). В процессоре NXP LPC55S6x реализован другой подход: в зависимости от режима безопасности процессоры имеют открытый или закрытый доступ к памяти и прочим ресурсам (см. рис. 7).

Таким образом, физический блок памяти разделен на безопасную и небезопасную области.

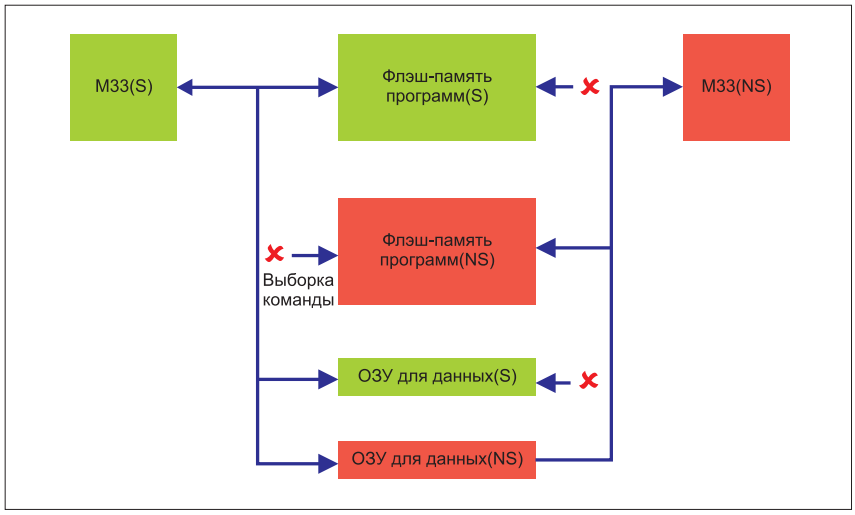

Рис. 7. **Процессоры NXP LPC55S6x обеспечивают безопасный режим работы ядра (состояние S) при чтении защищенных областей памяти программ (показаны зеленым) и небезопасный режим (NS) при чтении с незащищенной памяти программ (показаны красным)**

ПОДКЛЮЧЕНИЕ К ОБЛАКУ

При подключении к облаку или удаленному процессору потенциально возникает угроза безопасности. Во-первых, расширяется сеть. В нее могут проникнуть вредоносные сервисы под видом проверенных. В то же время, облако также подвергается угрозе, когда к нему обращается вредоносное ПО под видом узла IoT. Для защиты обеих сторон используются специальные протоколы безопасности, обеспечивающие проверку пользователя и сервиса.

Например, сервисы AWS (Amazon) и Azure (Microsoft) обеспечивают входной портал для взаимодействия между устройством и полным набором облачных сервисов (виртуальная машина, ПО как услуга (SaaS) и др.). Между облаком и предприятием используется аналогичный портал. В нем применяются протоколы аутентификации, выполненные с помощью соответствующих пакетов разработки. Например, в AWS задействован шлюз устройства (см. рис. 8).

При подключении к облаку на стадии аутентификации устройство должно предоставить приватный ключ, сертификат X.509 или другой идентификационный параметр (в зависимости от типа облака). Облачные сервисы также запра-

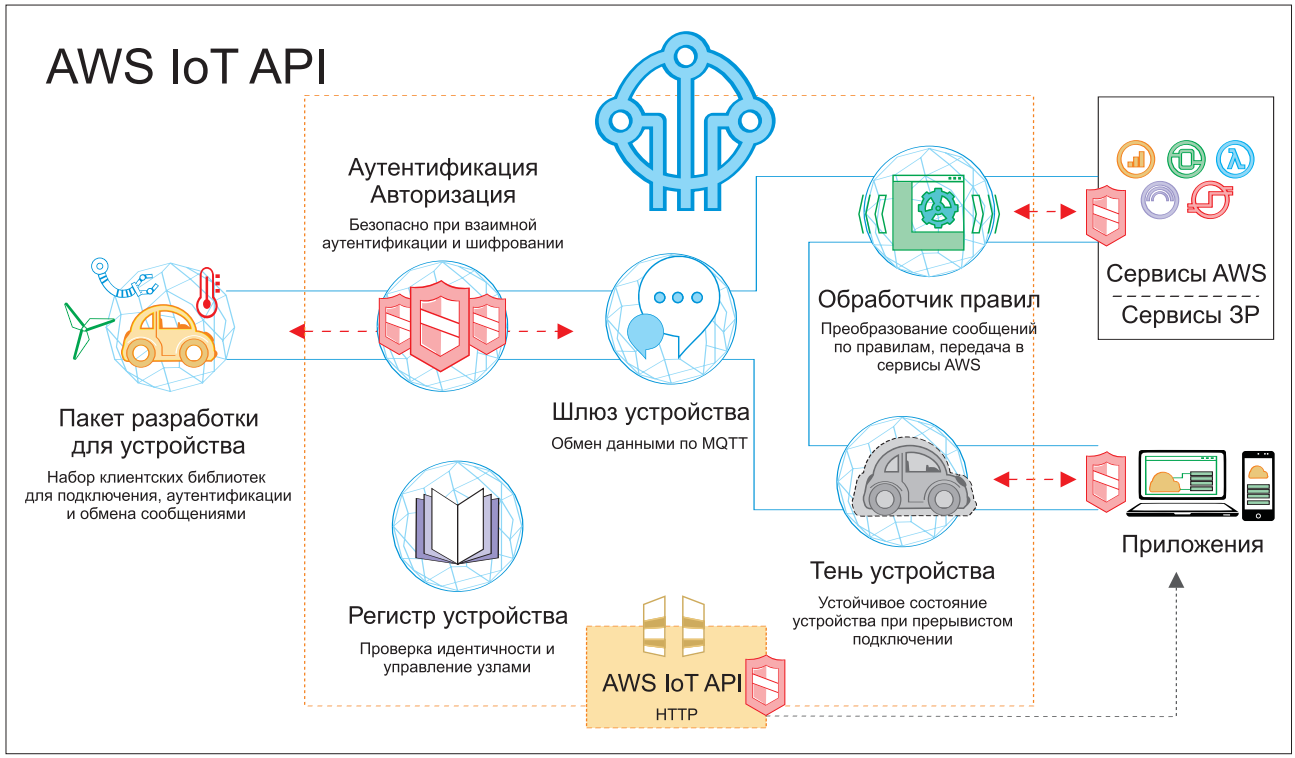

Рис. 8. **Как и прочие поставщики облачных услуг, AWS предоставляет ряд сервисов, обеспечивающих безопасность и эффективность проведения транзакций между узлами IoT и облаком**

шивают спецификацию политик для определения прав доступа устройства к облачным сервисам.

При неправильной настройке процедуры аутентификации или распределения прав могут появляться лазейки, через которые злоумышленник атакует устройство, сеть или приложение. Готовые средства разработки и пакеты ПО позволяют упростить процедуру подключения к облаку и избежать типичных ошибок. Их выпускают не только поставщики облачных сервисов, но и производители электронных компонентов, например Microchip или Renesas.

УПРОЩЕНИЕ ПРОЦЕДУР

Поскольку не всегда требуются тщательные процедуры защиты, разработаны методы их автоматизации. Например, AWS IoT использует сертификаты начальной загрузки (см. рис. 9).

Имея такой сертификат, устройство подключается к облаку с минимальным количеством прав (см. стадию 1 на рисунке 9), запрашивает новый сертификат (2), получает URL сертификата (3), сгенерированный инструментом AWS Lambda, и отзывает этот сертификат из корзины AWS Simple Storage Services (4). С новым сертификатом устройство подключается к AWS IoT (5) и начинает полноценную работу с облаком.

Однако за безопасность работы с облачными сервисами отвечает не только их поставщик, но и пользователи (см. рис. 10). Они ответственны за приложения, данные и ресурсы в облаке.

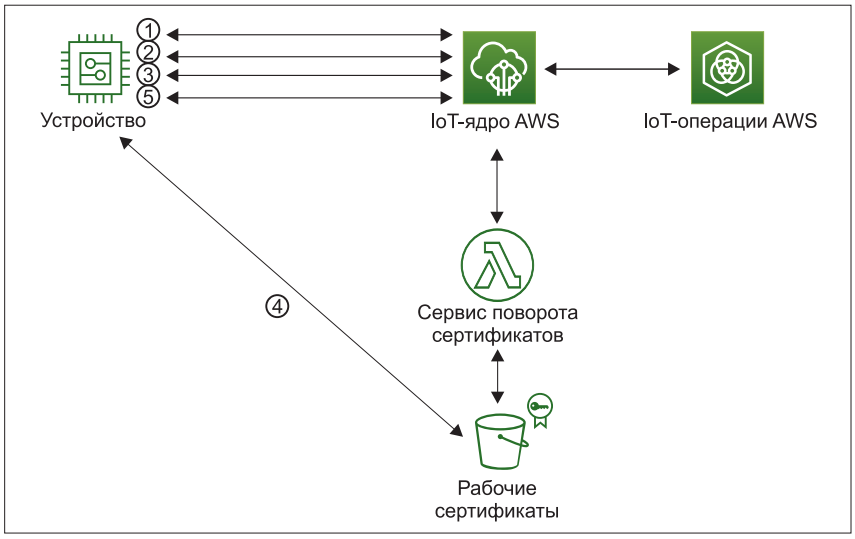

Рис. 9. **Облако AWS IoT использует сертификаты начальной загрузки для подключения IoT-узлов**

ВЫВОДЫ

Итак, проблема защиты интернета вещей является чрезвычайно важной. Необходимо обеспечить безопасность каждого узла, причем на нескольких уровнях, чтобы свести к минимуму возможности несанкционированного доступа. Существует немало разных протоколов безопасности, большинство из которых основано на использовании секретных кодов того или иного типа. Механизмы защиты также встроены в криптопроцессоры и микроконтроллеры.

Кроме того, следует помнить, что защита обеспечивается на всех этапах работы: при аутентификации, функционировании, перезагрузке и т. д. При использовании облачных сервисов или удаленных серверов возникает проблема защищенности этих внешних ресурсов от потенциально опасных пользователей. Причем, речь идет не только о защите каналов передачи, но и конечных узлов, т. е. самого облака и устройства, к нему обращающегося.

Таким образом, в настоящее время представлен полный спектр программно-аппаратных средств, обеспечивающих безопасность работы. По-прежнему ключевым требованием является многоступенчатость проверок и разнообразие механизмов защиты, делающих атаки на систему нецелесообразными и дорогостоящими.

Рис. 10. **Разделение ответственности за обеспечение безопасности между облаком и узлом**