Алгоритм аутентификации и генерации общего сессионного ключа симметричного шифрования для устройств Интернета вещей на основе протокола MQTT

The algorithm for authentication and generation of a shared session key of symmetric encryption for the IoT device based on the MQTT Protocol

Дикий / Dikii D.
Дмитрий Игоревич
(dimandikiy@mail.ru)
ФГАОУ ВО «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»,
аспирант факультета безопасности информационных технологий.
г. Санкт-Петербург

Ключевые слова: аутентификация – authentication; MQTT; без разглашения – non-disclosure; управление ключами – key management; Интернет вещей – Internet of things; симметричное шифрование – symmetric encryption; межмашинное взаимодействие – machine-to-machine interaction.

В данной статье описан алгоритм аутентификации устройств межмашинного взаимодействия по протоколу MQTT, который позволяет удостовериться в легальности устройства без передачи пароля в явном виде, а также его хеш-значения . Представлен сам алгоритм, сферы его применения, ограничения, которые накладывает алгоритм. Проведен эксперимент по сравнению с протоколом TLS, получены результаты, показывающие заметное улучшение быстродействия по сравнению с протоколом TLS. Криптостойкость предлагаемого алгоритма основана на проблеме дискретного логарифма. Основным преимуществом алгоритма перед протоколом TLS является возможность аутентификации за один такт «запрос–ответ», что позволяет использовать его совместно с протоколом MQTT, а не создавать защищенный канал на других уровнях сетевого взаимодействия. Одним из наиболее важных ограничений алгоритма является обязательная синхронизация устройств по времени.

This article describes the algorithm for authentication of machine-to-machine interaction devices using the MQTT Protocol, which makes it possible to verify the legality of the device without passing the password explicitly, as well as its hash value. The algorithm itself, its scope and restrictions imposed by the algorithm are presented. An experiment was conducted in comparison with the TLS Protocol, the results showing a noticeable improvement in fast operation compared to the TLS Protocol. The cryptographic security of the proposed algorithm is based on the discrete logarithm problem. The main advantage of the algorithm over the TLS Protocol is the ability to authenticate in one "request-response" step, which makes it possible to use it in conjunction with the MQTT Protocol, and not to create a secure channel at other network interaction levels. One of the most important limitations of the algorithm is the mandatory timing of devices.

Введение

С развитием парадигмы Индустрия 4.0 и все большим внедрением киберфизических систем в повседневную жизнь человека актуальность набирают вопросы безопасности внедряемых технологий. Одним из наиболее популярных решений в области автоматизации является использование технологий «Интернета вещей». Сюда входят «Умный дом», «Умный город» и множество других разработок [1]. Основными отличиями используемого оборудования в Интернете вещей от обычного являются не только способность к принятию решений на основе входных данных и коммуникационное соединение по сети Интернет, но и ограниченность в вычислительной мощности, запасе энергии автономного источника электропитания. Эти аспекты накладывают множественные ограничения. Для разрешения проблемы экономичной передачи данных между сотнями и тысячами устройств разрабатываются различные легковесные протоколы коммуникации. Так, в 2013 году миру был представлен алгоритм *CoAP* [2], по своей архитектуре напоминающий протокол *HTTP*, протокол *MQTT*, реализующий модель «издатель-подписчики» [3], протокол *XMPP* [4]

и *DDS* [5]. Все эти протоколы работают поверх стека протоколов *TCP/IP*. Другое направление развития интернета вещей – это разработка протоколов более низких уровней модели *OSI*. Наиболее известные из них: стек протоколов *ZigBee* и протокол *LoRaWAN* [1,6]. Обеспечение защиты данных в перечисленных выше протоколах организовано по-разному. Из механизмов безопасности во всех протоколах присутствует контроль целостности в виде протокола *CRC* (*Cyclic Redundancy Code*). Контроль конфиденциальности производится за счет симметричного алгоритма шифрования *AES*-128 бит (*Advanced Encryption Standard*) [7–9]. Протоколы более высоких уровней обеспечивают ассиметричное и симметричное шифрование передаваемых данных, аутентификацию пользователей информационного обмена, контроль целостности [10]. Как правило, функции безопасности требуют дополнительных вычислительных, энергетических затрат, также задаются более высокие требования к постоянной и энергозависимой памяти исполнительных устройств. В данной статье описывается алгоритм, который позволяет проводить аутентификацию устройств для межмашинного взаимодействия без передачи ключевой информации по открытым каналам связи, сгенерировать общий сессионный ключ для последующего симметричного шифрования блоков данных. Данный протокол предполагает полную совместимость с одним из наиболее широко распространённых протоколов среды Интернета вещей – *MQTT*, что позволяет сравнить предлагаемый алгоритм с уже существующими методами защиты информации.

Обзор

Согласно обзору, представленному в работе [11], ботнет-сеть *Mirai* бросила вызов всемирному обществу с точки зрения кибератак. Устройства получали неавторизованный доступ к другим устройствам и выполняли на нем зловредный код. Чтобы избежать подобных инцидентов, авторы выделили основные механизмы безопасности, которые должны быть реализованы. Одной из наиболее важных была объявлена аутентификация. Таким образом, разработка протокола аутентификации приспособленного к среде Интернета вещей является актуальной. Применительно к протоколу *MQTT* часто используется протокол *TLS*, и ведутся работы по разработке протокола *OAUth* 2.0 [12]. Другим направлением в области аутентификации устройств является разработка алгоритмов на базе алгоритма нулевого знания, представленного Фиатом и Шамиром [13]. В работе [14] авторы предлагают алгоритм для межмашинной аутентификации, где создается граф устройств с таблицей их MAC-адресов в качестве ключевого поля, а генерация сессионного ключа происходит по алгоритму Диффи-Хелмана. Такой алгоритм подходит для архитектуры сети «каждый с каждым». Авторы работы [15] предлагают многоуровневую аутентификацию на базе алгоритма без разглашения. Этот метод также основан на построении графов, что требует неприемлемо большое количество раундов обмена сообщениями между устройствами. [16]. В работе [17] представлен иной подход к аутентификации. Здесь предлагается использовать ассиметричное шифрование на эллиптических кривых. Однако с вычислительной точки зрения это достаточно затратная операция. Авторы работы [18] предлагают использовать отдельный сервер аутентификации и авторизации, что не всегда применимо. В работе [19] представлен легковесный протокол аутентификации Интернета вещей, основанный на алгоритме симметричного блочного шифрования. Такой алгоритм больше подходит для аутентификации вида «точка-точка». Авторы работы [20] предлагают метод, основанный на токенах безопасности с привлечением сервера авторизации и аутентификации.

Алгоритм

Протокол *MQTT* (*Message Queue Telemetry Transport*) имеет архитектуру «издатель-подписчик», состоящую из двух основных компонентов. Первый – это сами оконечные устройства, которые осуществляют коммуникационный обмен. Вторым элементом является брокер или шлюз («*gate*» – англ.). В его задачи входит логистика входящих и исходящих сообщений. Таким образом, сообщение, поступающее на шлюз, перенаправляется одному или нескольким устройствам, для которых это сообщение предназначено. Данный алгоритм удобен при многоадресной рассылке. Так, устройство-«издатель» генерирует одно исходящее сообщение для *n* абонентов. Данная модель реализована с помощью такого понятия, как «Тема» или «Топик». Каждое сообщение маркируется темой. Чтобы получать сообщения с той или иной темой, устройство-«получатель» подписывается на эту тему.

Из средств обеспечения безопасности в данном протоколе предусмотрено два механизма: аутентификация по паре логин/пароль и контроль доступа с помощью локального *ACL* (*access list*) файла или базы данных. Пароль и логин передаются в открытом виде в теле сообщения *CONNECT* (протокол *MQTT* поддерживает 16 видов сообщений [3]). Если аутентификация пройдена, брокер отвечает соответствующим сообщением *CONNACK*.

Как представлено в работах [1, 11, 12, 21, 22], предлагается использовать протокол *TLS* для обеспечения безопасной аутентификации между устройствами и шлюзом. Данный протокол подразумевает две стадии: генерация общего сессионного ключа на основе сертификатов в формате X509 по протоколу Диффи-Хелмана (*DH – Diffie-Hellman* [23] или *ECDH – Elliptic curve Diffie-Hellman* [24]), использование алгоритма симметричного или ассиметричного шифрования, например, *AES*, *RSA*, *ChaCha*20, а также

хеш-функции, например, *SHA*256 [25]. Недостатки такого подхода заключаются в следующем. Во-первых, в цепочку включается третье лицо – удостоверяющий центр, который может подтвердить достоверность выданных сертификатов для устройств и шлюзов. Во-вторых, устройства обязаны хранить в энергонезависимой памяти свой сертификат, закрытый ключ, а также сертификат шлюза. В-третьих, шифронаборы (*cipher suites*), используемые в шлюзе, должны поддерживаться устройствами. При отсутствии такой совместимости невозможно будет создать безопасный канал передачи данных. Из достоинств данного метода можно выделить полное шифрование канала связи. Для подключения по протоколу *TLS* устройство должно обращаться по адресу шлюза, но с другим номером порта. Например, для незащищенного соединения адрес выглядит «tcp://127.0.0.1:1883», тогда для соединения по протоколу *TLS* – «ssl://127.0.0.1:8883». Пример взят из документации к шлюзу с открытым исходным кодом, который был использован в данной работе [28]. Таким образом, создается безопасное соединение по протоколу *TLS*, а уже затем происходит аутентификация на шлюзе с помощью проверки на соответствие пары логин/пароль. Стоит отметить, что использование ассиметричного шифрования более затратное с точки зрения вычислительных мощностей. Поэтому разработчики в среде Интернета вещей стараются минимизировать использование ассиметричной криптографии.

В данной статье описывается новый алгоритм аутентификации устройств среды Интернета вещей без разглашения пароля, привлечения третьих сторон и использования сертификатов безопасности. Данный алгоритм также должен быть применим для генерации общего сессионного ключа для последующего использования в симметричном блочном или потоковом шифровании. Основным его отличием от рассмотренных алгоритмов является возможность аутентификации за один такт «запрос-ответ». Это позволяет использовать аутентификацию средствами протокола *MQTT* и делает ее более экономичной с точки зрения затрат энергии. Также алгоритм должен быть совместим с существующими системами, не нарушая их функциональности.

Предлагаемый алгоритм основан на базе проколов без разглашения («*zero-knowledge protocol*») – алгоритма Шнорра [16] и алгоритма, представленного в работе [26]. Впервые алгоритм без разглашения был представлен в конце XX века. Основная цель такого алгоритма – убедиться серверу (шлюзу) в том, что

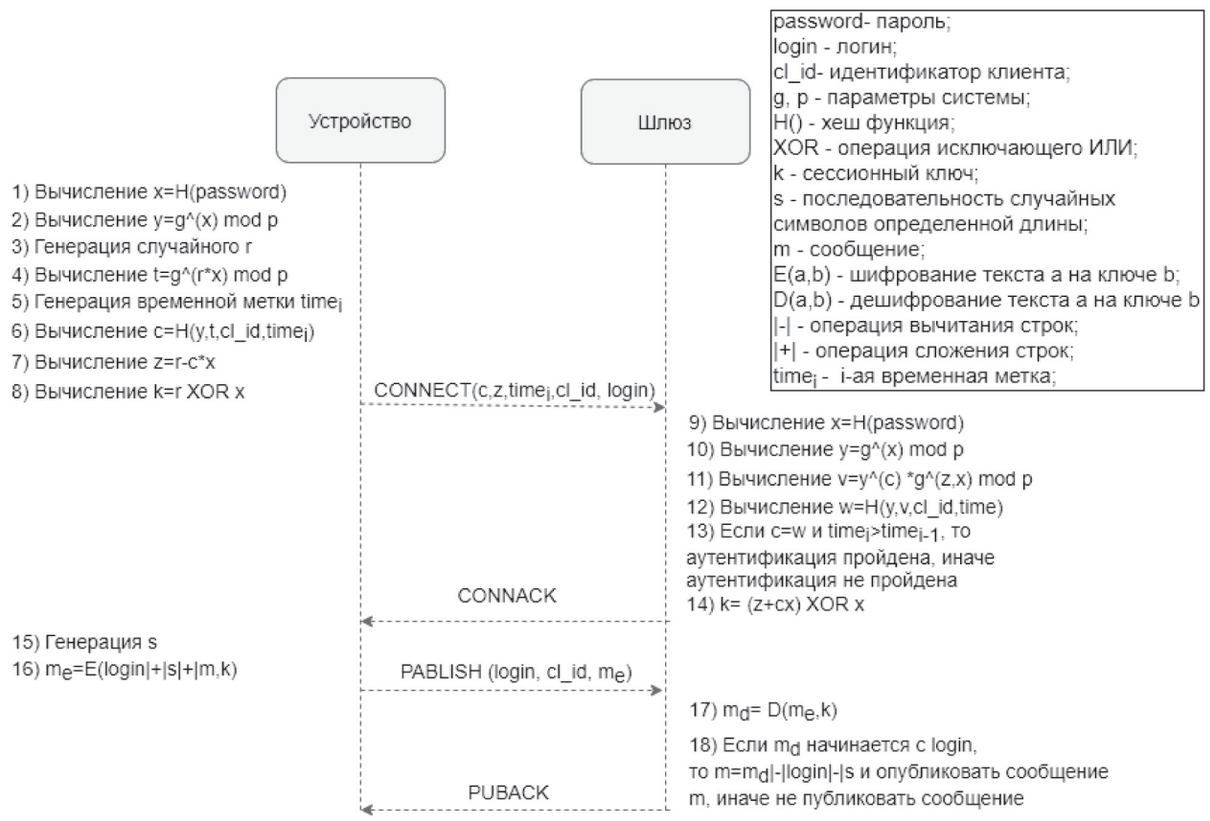

Рис. 1. Алгоритм аутентификации и генерации общего сессионного ключа без разглашения

клиент (устройство) знает пароль без непосредственной его передачи. Сложность использования этих протоколов применительно к протоколу *MQTT* обуславливается тем, что в нем для аутентификации используется всего лишь один такт «запрос-ответ».

Предлагаемый алгоритм представлен на рис. 1.

Представленная на рис. 1 схема описывает подключение клиента с помощью сообщения *CONNECT* к шлюзу и отправку одного сообщения *PUBLISH*. Шлюз посылает соответствующие ответные сообщения *CONNACK* и *PUBACK*. Вначале устройство-клиент вычисляет хеш-значение от пароля, например, с помощью функции *SHA*256. Затем вычисляет параметр y, при этом параметры p и g общедоступные и известны всем участникам. Как правило, p – это большое простое число, а g – такое число, что $g^q \bmod p=1$, где q – такое число, что $(p-1)\ mod\ q=1$. Затем генерируется случайное большое число r, которое будет использоваться, как гамма для генерации общего сессионного ключа. Вычисляется число t. Запоминается время подключения в миллисекундах – переменная $time_i$. Эта переменная необходима во избежание повторной аутентификации по одному и тому же сообщению *CONNECT*. Шлюз будет проверять значение временной метки, идентификатора клиента и логина позже. Если временная метка окажется неактуальной (сообщение с таким содержимым уже было получено ранее), устройству-клиенту будет отказано в аутентификации. Следующий шаг – это вычисление хеш-значения с от параметров y, t, идентификатора клиента cl_id и временной метки $time_i$. Вычисляется число z. Устройство-клиент отправляет сообщение *CONNECT* с содержимым в поле *username* – логин пользователя; в поле *password* – строка, состоящая из значений c, z, $time_i$. Данные об идентификаторе клиента всегда отправляются в служебном заголовке, поэтому нет необходимости дублировать информацию, но присутствие в хеш-значении c однозначно идентифицирует сообщение с устройством-клиентом. Это необходимо для защиты от подмены сообщений. Шлюз, получив входящее сообщение *CONNECT*, находит пользователя по логину, например в базе данных, вычисляет хеш-пароля, вычисляет значения y, w и сравнивает значение c из входящего сообщения с вычисленным значением w, а также проверяет актуальность временной метки. Если метка актуальная, то она запоминается. Генерация общего сессионного ключа производится путем наложения гаммы (число r для устройства-клиента и сумма $z+cx$ для шлюза). Формула (1) представляет собой перечень вычислений, производимых на стороне устройства-клиента, где x – хеш-значение пароля.

$$\begin{cases} y = g^x \\ t = g^{rx} \\ z = r - cx \end{cases} \quad (1)$$

На стороне брокера производятся следующие вычисления – формула (2):

$$\begin{cases} y = g^x \\ v = y^c g^{xz} \end{cases} \quad (2)$$

Подставим значения y и z в нижнее равенство формулы (2) и получим равенство, согласно формуле (3):

$$v = g^{xc} g^{x(r-cx)} = t \quad (3)$$

Раскрывая скобки, мы получим $v=g^{xr}$, что на стороне клиента определено, как переменная t. Таким образом, хеш-значение c, состоящее из значений y, t, $time_i$, cl_id возможно вычислить на стороне шлюза, как хеш-значение от y, v, $time_i$, cl_id.

На основании хеш-значения пароля x и случайно сгенерированного числа r вычисляется общий сессионный ключ для симметричного шифрования, например, для алгоритма *AES* 128бит. Для устройства-клиента достаточно вычислить значение побитного исключающего «или» между хеш-значением пароля x и числа r. На стороне шлюза значение r вычисляется согласно следующей формуле:

$$r = z + cx \quad (4)$$

Криптостойкость данного алгоритма основана на проблеме дискретного логарифмирования, которую невозможно решить за приемлемое время при достаточно больших значениях параметрах системы p, g.

Для обеспечения уверенности в том, что шлюз общается с тем устройством, которое аутентифицировано, в начало каждого последующего сообщения вида *PUBLISH* (отправить сообщение), *SUBSCRIBE* (подписаться на «Тему»), *UNSUBSCRIBE* (отписаться от «Темы») включается логин пользователя с добавлением случайной последовательности символов определенной длины, зашифрованный на общем сессионном ключе. Длина случайной последовательности должна быть согласована заранее, например в реализации алгоритма использована следующая методика: пока остаток деления длины логина на восемь не равен нулю, добавить в конец логина один случайный символ. Если длина логина кратна восьми символам, то добавить восемь случайных символов в конец логина. Это позволит получать различные варианты зашифрованного текста при блочном шифровании на одном и том же логине и общем сессионном ключе. Добавление логина в начало сообщения позволит шлюзу убедиться, действительно ли устройство обладает общим сессионным ключом. Если при дешифровании в начале сообщения не будет обнаружено значение логина устройства-клиента, то сообщение дальше обрабатываться не будет.

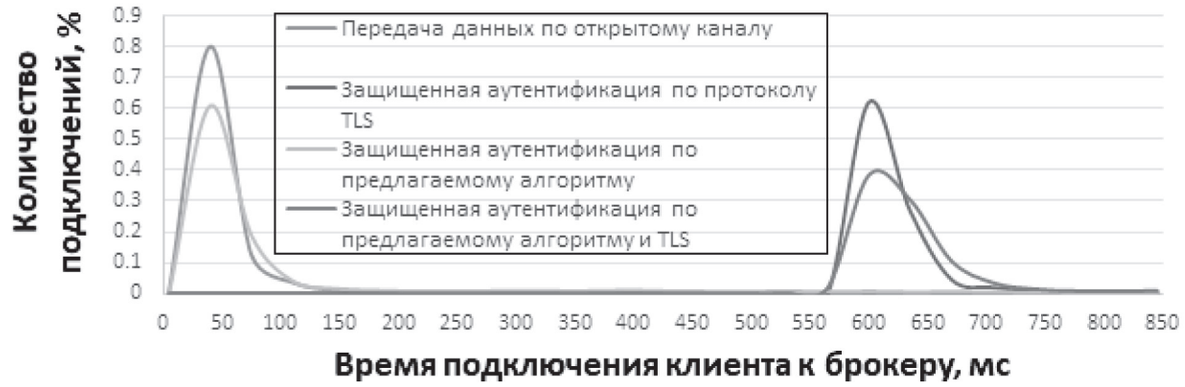

Рис. 2. График сравнения времени, затраченного на соединение по открытым и защищенным каналам связи. Ширина окна для построения графика – 35 мс

Такие механизмы, как добавление логина со случайной последовательностью символов в начало сообщения, добавление временной метки, а также аутентификации без разглашения, нивелируют угрозу реализации атаки вида «человек посередине».

Криптостойкость передаваемых сообщений зависит от криптостойкости блочных и потоковых шифроалгоритмов, поэтому стоит использовать наиболее криптостойкие алгоритмы. В данной работе использовался алгоритм блочного шифрования *AES* в режиме *ECB* – режиме простой замены. Анализ работы алгоритма блочного шифрования *AES* в зависимости от выбранного режима работы приведен в работе [27].

К преимуществам разработанного алгоритма можно отнести то, что по сравнению с протоколом *TLS*, аутентификация происходит за один такт «запрос-ответ», не требуется участие третьей стороны, отсутствует необходимость в хранении сертификатов в энергонезависимой памяти. К недостаткам можно отнести настройку устройства. Такие параметры системы, как p и g, должны быть известны перед подключением, что потребует дополнительной предварительной настройки устройства перед использованием. Алгоритм блочного/потокового шифрования также должен быть согласован заранее, так как использование различных алгоритмов приведет к неправильной работе алгоритма.

Сравнение алгоритма с протоколом TLS

Для сравнения предлагаемого алгоритма с протоколом *TLS* было произведен эксперимент по следующей методике. Был программно доработан *MQTT* брокер с открытым исходным кодом [28]. В качестве шлюза использовался микрокомпьютер *Raspberry pi 3 model B*, со следующими характеристиками: процессор *ARM Cortex-A*53 с тактовой частотой 1,2 ГГц, ОЗУ 1 Гб. Наиболее затратная операция – это подключение устройства к брокеру. Поэтому было произведены измерения затраченного времени на соединение по открытому каналу, каналу, защищенному по протоколу *TLS*, каналу, защищенному предлагаемым алгоритмом, каналу, защищенному по протоколу *TLS* и предлагаемому алгоритму одновременно. Эксперимент состоял из пяти тысяч подключений по каждому из четырех исследуемых направлений. Результаты эксперимента представлены на рис. 2.

В ходе эксперимента были получены статистические параметры о времени подключения устройств-клиентов к шлюзу. Данные представлены в таблице 1.

Согласно результатам эксперимента, можно сделать вывод о том, что аутентификация по открытому каналу связи самая быстрая, обладает наименьшим среднеквадратичным отклонением. Использование протокола *TLS* значительно замедляет процесс аутентификации. Так, среднее время подключения клиента к брокеру в 17 раз медленнее и составляет около половины секунды. Предлагаемый алгоритм обладает большим быстродействием: среднее время подключения относительно аутентификации по открытому каналу связи медленнее примерно в три раза (60% подключений происходило менее чем за 50 мс), и в пять раз быстрее, чем при использовании протокола *TLS*. Однако предлагаемый алгоритм обладает большей дисперсией и среднеквадратичным отклонением, что свидетельствует о разбросе статистических данных. Применение одновременно двух механизмов обеспечения защиты передаваемых данных при аутентификации показало наихудший результат – почти три четверти секунды на подключение.

Чтобы оценить влияние добавления логина со случайной последовательностью символов в начало каждого сообщения, было измерено время доставки сообщения по открытым и защищенным каналам. Эксперимент показал отсутствие явного влияния внедрения дополнительной информации на быстро-

Таблица 1

Сравнение времени подключения устройств к шлюзу в миллисекундах при использовании открытого и защищенных каналов связи

Вид аутентификации / Характеристика	Аутентификация по открытому каналу	Аутентификация по защищенному каналу по протоколу TLS	Защищенная аутентификация по предлагаемому алгоритму	Защищенная аутентификация по защищенному каналу по протоколу TLS и предлагаемому алгоритму
Минимальное время подключения, мс	5	535	12	548
Максимальное время подключения, мс	1319	3818	3072	9137
Среднее время подключения, мс	33.3	582.1	108.7	739.2
Среднеквадратичное отклонение	51.8	116.8	227.5	455.3

действие доставки сообщения, разница составила 1–2 мс.

Таким образом, показана эффективность предлагаемого алгоритма аутентификации применительно к среде Интернета вещей, где важную роль играют быстродействие и экономия энергии и вычислительных затрат.

Заключение

В данной работе представлен алгоритм аутентификации устройств межмашинного взаимодействия на базе протокола передачи данных *MQTT*, который позволяет удостовериться в подлинности устройства без непосредственной передачи пароля. Наиболее часто используемым способом защиты данных в протоколе *MQTT* является создание защищенного соединения по протоколу *TLS*. К недостаткам такого подхода можно отнести привлечение третьей стороны (удостоверяющего центра), хранение сертификатов в энергонезависимой памяти устройств, обязательная совместимость наборов криптографических алгоритмов на шлюзе и устройстве-клиенте, что вносит дополнительные требования к проектированию устройств. Представленный алгоритм обладает рядом преимуществ относительно протокола *TLS*, а именно:

· отсутствие передачи пароля от клиента к шлюзу;

· генерация общего сессионного ключа, как и в протоколе *TLS*;

· отсутствие необходимости в использовании и хранении сертификатов безопасности в формате X509;

· отсутствие зависимости от третьих сторон;

· более быстрое взаимодействие устройств, что доказано результатами проведения эксперимента;

· криптостойкость обеспечивается неразрешимостью за приемлемое время проблемы дискретного логарифма;

· защита от атак вида «человек посередине», путем использования временных меток и дополнительных данных в полезной нагрузке сообщения;

· полная совместимость с протоколом передачи данных *MQTT*;

· генерация общего сессионного ключа и аутентификация за один такт «запрос-ответ»;

· обеспечение конфиденциальности путем использования блочного алгоритма шифрования.

К недостаткам алгоритма можно отнести зависимость от синхронизации устройств по времени. Использование временных меток подразумевает, что время на устройстве должно быть синхронизировано и не может «идти назад». Также устройство должно хранить параметры системы (два больших простых числа p и g) в постоянной энергонезависимой памяти. При передаче сообщений образуется небольшая избыточность данных.

Литература

1. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications / A. Al-Fuqaha [et al.] // IEEE Communication surveys & tutorials. – 2015. – Vol. 17, No. 4. – P. 2347–2376.

2. ITU-T/ The Constrained Application Protocol (CoAP) / RFC 7252 – Proposed Standard. 2014.

3. ISO/IEC 20922:2016 Information technology – Message Queuing Telemetry Transport (MQTT) v3.1.1.

4. ITU-T/ Extensible Messaging and Presence Protocol (XMPP): Core // RFC-3920. 2011.

5. OMG/ DDS v1.4 – the DDS specification / Object Management Group. 2015.

6. Usman, R. Low Power Wide Area Networks: An Overview / R. Usman, K. Parag, M. Sooriyabandara // IEEE Communication surveys & tutorials. – 2017. – Vol. 19, No. 2. – P. 855–873.

7. Gomez, C. Wireless home automation networks: A survey of architectures and technologies / C. Gomez, J.Paradells // IEEE Commun. Mag. – 2010. – No. 48. – P. 92–101.

8. Katsikeas, S. A lightweight and secure MQTT implementation for Wireless Sensor Nodes / S. Katsikeas. – Chania: Technacal university of Crete, 2016.— 137 p.

9. Granjal, J. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues / J. Granjal, E. Monteiro, J. Sá Silva // School of electronic & computer engineering. – 2016.

10. Internet of Things Communication Reference Model / A.H. Alhamedi [et al // IEEE Sixth international Conference on Computational Aspects of Social Networks (CASoN), 2014. – P. 61–66.

11. The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices/ G. Perrone [et ak.] // Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017), 2017. – P. 246–253.

12. Federated Identity and Access Management for the Internet of Things / P. Fremantle [et al // International Workshop on Secure Internet of Things. 2014. – P. 10–17.

13. Fiat, A. How to prove yourself: Practical solutions to identification and signature problems., In A. Odlyxko, editor, Advances in Cryptology/ A. Fiat, A. Shamir // Proc. of Crypto'86 (Lecture Notes in Computer Science 263), Springer-Verlag, 1987. Santa Barbara, Califomia, U.S.A., August 11–15. – P. 186–194.

14. Schukat, M. Zero-Knowledge Proofs in M2M Communication / M. Schukat, P. Flood // ISSC 2014 / CIICT 2014. – Limerick. 2014. – P. 1–5.

15. Multi-graph Zero-knowledge-based Authentication System in Internet of Things/ I-Hsun Chuang [et al.] // IEEE ICC 2017 SAC Symposium Internet of Things Track, 2017. – P. 1–6.

16. Введение в криптографию / под общ. ред. В. В. Ященко. – 4-е изд. доп. – М.: МЦНМО, 2012. – 348 с.

17. Secure MQTT for Internet of Things (IoT) / M. Singh [et al.] // 2015 Fifth International Conference on Communication Systems and Network Technologies. – P. 746–751.

18. Authorization Mechanism for MQTT-based Internet of Things / A. Niruntasukrat [et al.] // IEEE ICC2016-Workshops: W07-Workshop on Convergent Internet of Things, 2016. – P. 1–6.

19. Nan, J.-H. A Lightweight Authentication Mechanism between loT Devices /J-H. Han, J.-N. Kim // ICTC 2017. – P. 1153–1155.

20. Bhawiyuga, A. Architectural Design of Token based Authentication of MQTT Protocol in Constrained IoT Device / A. Bhawiyuga, M. Data, A. Warda// IEEE. – P. 1–4.

21. Lavinia, N. Security in the Internet of Things: A Survey on Application Layer Protocols / N. Lavinia// 21st International Conference on Control Systems and Computer Science, 2017. – P. 659–666.

22. Internet of Things: Survey and open issues of MQTT Protocol / M.B. Yassein [et al.] // ICEMIS2017, 2017. – P. 1–5.

23. Diffie,W. New Directions in Cryptography / W. Diffie, M.E. Hellman // IEEE Trans. Inf. Theory / F. Kschischang – IEEE. – 1976. – Vol. 22, Iss. 6. – P. 644–654.

24. Certicom Corp / Standards for Efficient Cryptography Group (SECG). SEC 1: Elliptic Curve Cryptography v. 2.0. 2009.

25. Венедюхин А. Ключи, шифры, сообщения: как работает TLS [Электронный ресурс] / А. Венедюхин. – Режим доступа: https://tls.dxdt.ru/tls.html#ecdsa, свободный. – Загл. с экрана.

26. Jun, L. J. Implementing Zero-Knowledge Authentication with Zero Knowledge (ZKA_wzk) / L.J. Jun, B. Temacek // The Python Papers Monograph 2: 9 Proceedings of PyCon Asia-Pacific, 2010. – P. 1–19.

27. Lightweight & Secure Industrial IoT Communications via the MQ Telemetry Transport Protocol/S. Katsikeas [et al.] // IEEE Symposium on Computers and Communications (ISCC), 2017. – P. 1–8.

28. Moquette Java MQTT lightweight broker [Электронный ресурс]. – Режим доступа: https://github.com/andsel/moquette [дата обращения: 21.10.2018], свободный. – Загл. с экрана.