УДК 004.056

ЗАЩИТА СООБЩЕНИЙ МЕЖДУ СЕРВЕРОМ И ПРИБОРАМИ ИНТЕРНЕТА ВЕЩЕЙ

© 2019 И. Е. Пустыльник, Ю. П. Преображенский

Воронежский институт высоких технологий (г. Воронеж, Россия)

Существует разновидность приборов интернета вещей, для которой нельзя применять обычные системы защиты информации ввиду малой программной и процессорной мощности данных устройств. Для данных устройств необходима программная поддержка, которая основана на использовании аппликационного протокола CoAP и физического протокола ZigBee. Наряду с этим для подобных приборов необходимо использовать протоколы шифровки/расшифровки сообщений, которые используют новейшие алгоритмы защиты AES/SHA256 и их легковесную реализацию типа TinyAES.

Ключевые слова: AES, TinyAES, CoAP, ZigBee.

Интернет вещей состоит из большого количества умных приборов, каждый из которых заряжен на выполнение определенных функций. При рассмотрении интернета вещей обычно выделяют две группы приборов, которые отличаются различным уровнем автономии.

- К первой группе относятся приборы с полноценной операционной системой, обычно на основе Linux. Такие приборы могут выполнять множество различных функций и являются существенно автономными (телефон, телевизор).
- Ко второй группе относятся приборы с существенно «усеченной» программной начинкой, которые могут выполнять минимальное количество операций, обычно связанных с определенной функцией (лампа, датчик).

У первой группы приборов важна, безусловно, начинка, которая выполняет различные операции и может быть каким-то образом перенаправлена на выполнение иных задач. Связь с этой группой устройств может осуществляться при помощи обычных сетевых протоколов, а именно HTTP/HTTPS, SOAP, JSON и так далее. Обычно такая связь осуществляется по каналу, который зашифровывается целиком (HTTPS) и позволяет передачу сообщений как при наличии, так и при отсутствии отдельного шифрования сообщений.

Приборы второй группы требуют несколько иного подхода к шифровке сообщений. Здесь уместно добавить, что данные приборы не имеют расширенного программного обеспечения. Таким образом, перехват сообщений, идущих к прибору или от него, фактически обозначают контроль за данным прибором. Поэтому очень важно выбрать алгоритмы шифрования сообщений, которые позволили бы обеспечить полную безопасность потока информации, идущего к прибору и от него. Еще одним очень важным фактором является объем и мощность алгоритмов шифрования, которая не вела бы к существенной потере скорости передачи сообщений и позволяла бы обработать их максимально быстро и без потери информации.

Одним из самых распространенных заблуждений разработчиков систем коммуникации в интернете вещей является принцип тождественности этих сетей и сетей обычного интернета. Для последних используется Инфраструктура Открытых Ключей (ИОК). Ее главное достоинство заключается в том, что ключ является не только средством чтения, но и предметом идентификации передающего сообщения. В традиционных сетях, которые полностью открыты для обзора третьим лицом (man in the middle) такой метод шифрования сообщений себя абсолютно оправдывает.

Если же рассмотреть ситуацию в интернете вещей, то сразу же бросится в глаза тот факт, что обычно сервер и все его клиенты друг другу известны. Все сообщения являются полностью ожидаемыми. Используя обычный вариант ИОК, для зашифровки сообщений, разработчики открывают весь парк ключей в течение очень короткого интервала времени. Если учесть, что ИОК не

Пустыльник И. Е. – Воронежский институт высоких технологий, студент.

Преображенский Юрий Петрович – Воронежский институт высоких технологий, к. т. н., профессор, petrovich@vivt.ru.

может передавать сообщения более 2048 байтов в обычном режиме шифровки, то использование данной системы может быть существенно затруднено.

Примерно, обмен сообщениями в системах интернета вещей может выглядеть так, как представлено на рисунке 1.

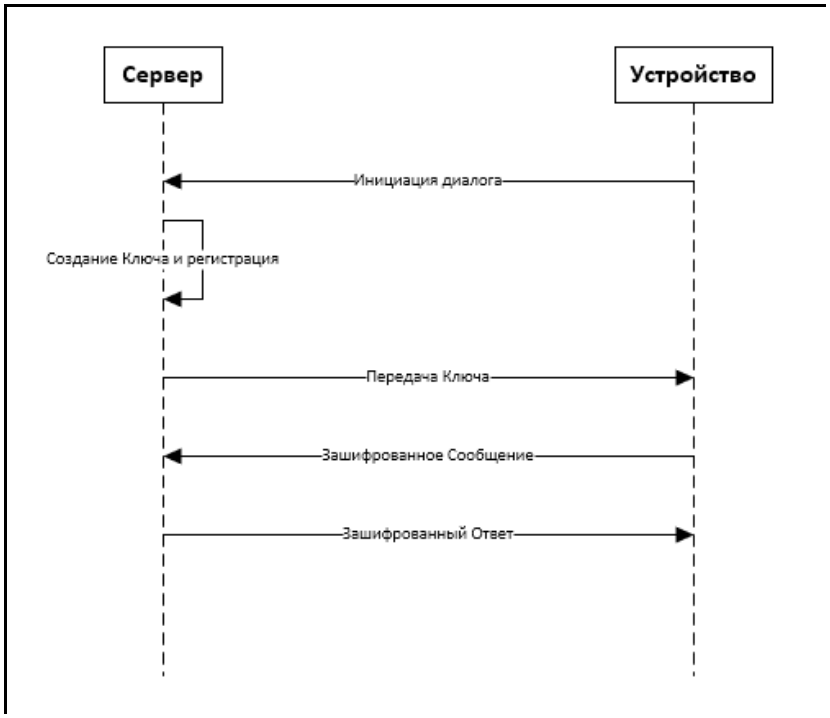

Рисунок 1. Примерная схема передачи сообщений в интернете вещей.

На первый взгляд, система передачи сообщений очень сходна с той, что используется в обычном интернете для передачи HTTPS-сообщений. Однако, существует одно очень существенное отличие. В обычном интернете на один сервер приходится практически неограниченное количество клиентов, которые доверяют не серверу, а его ключу шифровки сообщений, который покупается у провайдера ключей, которому, в свою очередь, доверяют клиенты, или точнее их браузеры.

В случае, показанном на рисунке 1, все клиенты знают о сервере, и сервер знает о наличии всех клиентов. В данном случае, когда клиент известен и может идентифицировать себя, становится возможным создание ключей «под клиента». Такие ключи могут передаваться от сервера к клиенту для определенного диалога и уничтожаться после того, как диалог между сервером и клиентом завершен. Перехват подобных сообщений при помощи атаки man-in-the-middle практически невозможен, потому что сообщения передаются при помощи сгенерированного одноразового ключа.

Существует гипотетическая возможность перехвата ключа до того, как обмен сообщениями состоялся. Таким образом, хакер может попытаться продублировать сообщения от клиента с перехваченным ключом. Для защиты от подобного гипотетического события используют дополнительную защиту информации с помощью случайно генерируемого номера «Salt», который вставляется в сообщение тоже произвольно. Такой номер может быть просто случайно сгенерирован для каждого клиента. Маловероятно, что хакер, не имеющий доступа непосредственно к прибору, сможет им воспользоваться.

Рассматривая ситуацию с небольшой сетью из маломощных клиентов, передающих информацию на сервер, можно также воспользоваться несколько иным протоколом передачи информации между клиентом и сервером. В настоящее время, существует протокол CoAP, который активно продвигается консорциумом IETF, как протокол, подходящий для передачи сообщений в интернете вещей. На рисунке 2 представлена сравнительная схема уровней OSI для протоколов CoAP и HTTP.

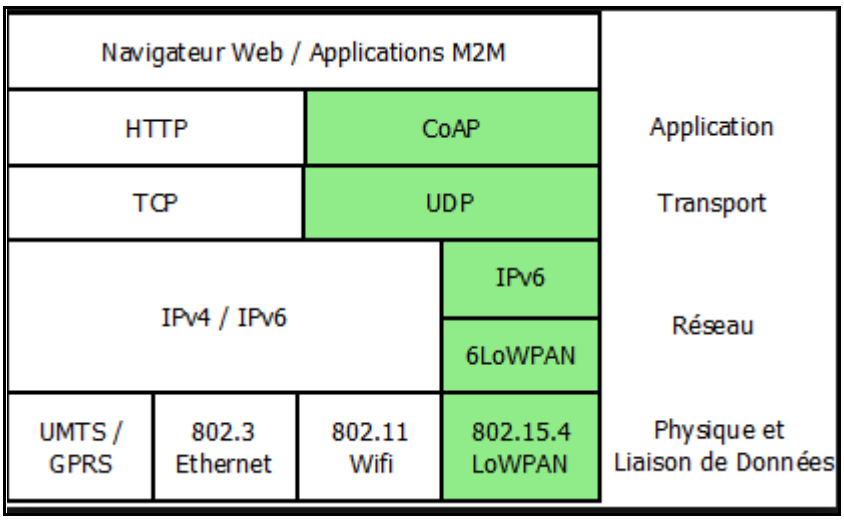

Рисунок 2. Сравнение протоколов CoAP и HTTP.

Из представленной диаграммы видно, что CoAP, как и HTTP находится на уровне протокола аппликаций. Существенным различием является то, что CoAP использует UDP в качестве транспорта. Это означает, что вся синхронизация сообщений идет на уровне аппликаций, что убыстряет передачу в обычном случае. На более низком уровне может использоваться как IPv6 в случае проводных сетей, так и 6LoWPAN для беспроводной передачи от устройств интернета вещей. Опять же, для физической передачи сигналов используется стандарт 802.15.4, который существенно «облегчен» по сравнению с Wi-Fi.

Еще одно преимущество протокола CoAP – облегченный характер сообщений. По сравнению с обычным пакетом, используемым при передаче сообщений в HTTP, заголовок протокола CoAP существенно меньше. Рисунок 3 показывает полную побайтовую разбивку CoAP.

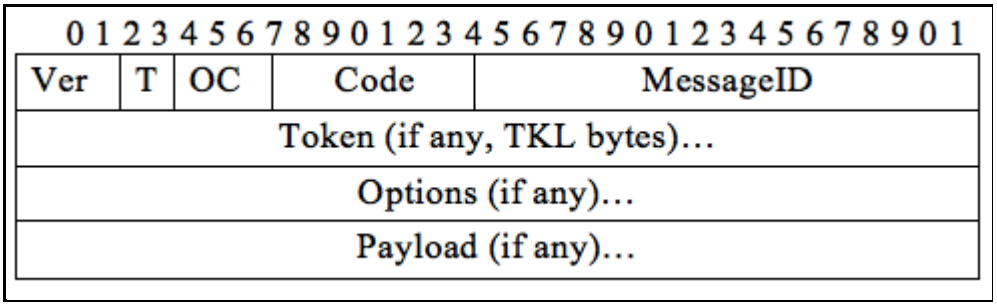

Рисунок 3. Побайтовая разбивка протокола CoAP.

Очевидно, что минимальное сообщение по протоколу CoAP составляет 32 байта. Такое практически невозможно ни в одной из традиционных сетей, используемых для передачи информации в формате HTTP.

Противники данного протокола могут возражать против его использования в сетях, требующих надежной передачи сообщений. В данном случае речь идет о сетях интернета вещей. Существуют два возможных способа избежать потери информации при передаче ее с помощью протокола UDP:

- Можно считать, что потерянное сообщение не несет уникальной информации и следующее сообщение восстановит информационную картину целиком
- Если сообщение достаточно важно для полной информационной картины, то подтверждение о получении сообщения должно быть частью прокола обмена информацией.

Таким образом, если необходимая картина передачи сообщений полностью зашита в протокол обмена, то UDP не будет созда-

вать существенных проблем для системы объектов интернета вещей.

Еще одним препятствием к использованию протокола CoAP можно считать то, что он технически опирается на сетевой формат ZigBee. Использование данного протокола требует также использования моста или Gateway между системами интернета вещей и обычной сетью, которая будет обрабатывать приходящую информацию. Несмотря на кажущиеся сложности, данная система потребует от хакера существенных усилий для организации взлома, т. к. часть системы будет находиться за Gateway и потребует дополнительного доступа к ней. С другой стороны, способы передачи сообщений между Ethernet и ZigBee хорошо отработаны и не потребуют никаких дополнительных инженерных затрат.

Кроме создания сети и протокола передачи сообщений, создатели любой сети с участием интернета вещей должны обратить внимание на инструменты зашифровки сообщений, передаваемых по данной сети. Ранее в данной статье говорилось о возможной громоздкости сообщений, которые передаются с помощью ИОК. Единственной альтернативой ИОК является шифрование сообщений с помощью симметричных алгоритмов.

В настоящее время используются алгоритмы DES, 3DES и AES, различающиеся размером ключа и размером блока сообщения. Несмотря на то, что алгоритм DES с ключом 56 битов практически не используется в интернете из-за возможности вскрытия сообщений с помощью грубой силы, его можно было бы использовать для связи систем интернета вещей. К сожалению, большинство библиотек, используемых вместе с данным алгоритмом, уже выведены из активного применения. Найти подобные библиотеки практически невозможно.

Единственной альтернативой DES является алгоритм AES, который используется с ключами трех различных размеров: 128, 192 и 256 бит. Алгоритм в данное время используется практически повсеместно в интернете и его надежность неоднократно проверена практикой. Существует набор компаний-провайдеров, которые занимаются выпуском ключей, которые можно использовать на интернет-серверах. Эти ключи пользуются полным доверием потребителей и изготовителей интернет-браузеров.

В случае интернета вещей и, соответственно, использования алгоритмов шифрования, можно воспользоваться тем же подходом, который представлен на рисунке 1. В настоящее время любое программное обеспечение, используемое на серверах, а именно операционные системы Windows и Unix (Linux) обладают необходимым программным обеспечением и надежными генераторами случайных чисел, требуемыми для выпуска подобных ключей.

К сожалению, использование стандартных библиотек, поддерживающих алгоритм шифровки/расшифровки AES на устройствах интернета вещей практически неосуществимо. Библиотеки типа .NET, openssl и другие требуют слишком большого объёма памяти и дискового пространства для использования в системах интернета вещей. Для работы на «умных» датчиках, «умных» лампах и других устройствах с ограниченным объёмом памяти, нужно использовать библиотеки, предоставляющие только набор базовых функций шифровки и расшифровки.

Одной из таких библиотек является TinyAES, которая находится в свободном доступе и может быть использована без ограничений. Авторы пакета утверждают, что он занимает не более 200 байтов оперативной памяти. Написанный на языке C, данный пакет может использоваться практически на любом устройстве, поддерживающем среду, в которой программа скомпилирована. Принимая во внимание то, что большинство встроенных компьютерных систем, используемых в «умных» приборах написаны на языке C, можно с уверенностью сказать, что библиотека, написанная на C может быть легко встроена в любую из вышеупомянутых систем.

Рассмотрение функций данного пакета показывает, что он поддерживает три основных режима AES, а именно ECB, CBC и CTR. Эти режимы позволяют различное сцепление блоков сообщения, когда оно шифруется. Соответственно, при расшифровке должен использоваться тот же самый режим, иначе сообщение не будет точно расшифровано. Такая проблема встречается в интернете вещей гораздо реже, т. к. и сервер, и умные устройства обычно производятся одним и тем же производителем.

```
/* Initialize context calling one of: */
void AES_init_ctx(struct AES_ctx* ctx, const uint8_t* key);
void AES_init_ctx_iv(struct AES_ctx* ctx, const uint8_t* key, const uint8_t* iv);

/* ... or reset IV at random point: */
void AES_ctx_set_iv(struct AES_ctx* ctx, const uint8_t* iv);

/* Then start encrypting and decrypting with the functions below: */
void AES_ECB_encrypt(struct AES_ctx* ctx, uint8_t* buf);
void AES_ECB_decrypt(struct AES_ctx* ctx, uint8_t* buf);

void AES_CBC_encrypt_buffer(struct AES_ctx* ctx, uint8_t* buf, uint32_t length);
void AES_CBC_decrypt_buffer(struct AES_ctx* ctx, uint8_t* buf, uint32_t length);

/* Same function for encrypting as for decrypting in CTR mode */
void AES_CTR_xcrypt_buffer(struct AES_ctx* ctx, uint8_t* buf, uint32_t length);
```

Рисунок 4. Листинг входов пакета TinyAES.

Так как в данной статье рассматривается защита связи между элементами интернета вещей и сервером, то необходимо отметить, что помимо защиты сообщений с помощью алгоритма AES, можно также использовать другие элементы защиты. Сервер может идентифицировать клиентов своей сети с помощью генерации и хранения специальных идентификаторов, которые генерируются с помощью алгоритма SHA256. Этот алгоритм тоже находится в открытом пользовании и поддерживается большинством библиотек шифровки/дешифровки сообщений. Если создавать на сервере так называемое хэш-слово, то его надо поддерживать с помощью других вспомогательных методов, например с помощью добавления секретного числа «Salt», упоминаемого ранее.

Использование подобных алгоритмов шифрования, типа AES, и алгоритма связи, показанного на рисунке 1, требует постоянной смены ключей, используемых в шифровке сообщений. В протокол связи между клиентом и сервером должна быть внесена модификация, позволяющая создавать ключи при инициации коммуникаций и определении клиента. По окончании диалога клиентом или по достижению таймаута, сервер должен удалить ключ или пометить его как использованный. Дальнейшее использование данного ключа должно быть невозможным.

В данной статье были рассмотрены различные варианты соединения и диалога между сервером и «легковесными» приборами интернета вещей. Очевидно, что в настоящее время данные приборы практически не унифицированы. Некая унификация существует на уровне физического протокола коммуникаций, например ZigBee. Унификации на уровне аппликаций пока нет. Рекомендации по шифровке сообщений могут быть применены отдельно к приборам, изготовленным в одной и той же фирме. Унификация протоколов и диалогов между приборами от различных производителей пока практически невозможна.

ЛИТЕРАТУРА

1. Дукас Х., Маглоджианнис И. и Флора В., «Enabling Data Protection through PKI encryption in IoT m-Health Devices» в 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE), Larnaca, Cyprus, 2012.

2. Арпторп Н., Райзман Д., Сундарешан С., Нараянан А. и Фимстер Н. «Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic» 16 08 2017. [В Интернете]. Available: https://arxiv.org/abs/1708.05044. [Дата обращения: 21 12 2018].

3. Доранс Б. Beginning ASP.NET Security, Glazgow, Great Britain: John Wiley and Sons Ltd., 2010.

4. Wikipedia, «CoAP» 7 12 2018. [В Интернете]. Available: https://fr.wikipedia.org/wiki/CoAP?uselang=fr. [Дата обращения: 21 12 2018].

5. Ш. Чен, «Constrained Application Protocol for Internet of Things» 30 04 2014. [В Интернете]. Available: https://www.cse.wustl.edu/~jain/cse574-14/ftp/coap/index.html. [Дата обращения: 21 12 2018].

6. Укил А., Бандиопадхяй С., Бхаттачария А., Пал А. и Босе Т. «Lightweight security scheme for IoT applications using CoAP» International Journal of Pervasive Computing and Communications, т. 10, № 4, pp. 372-392, 2014.

7. Kokke «Tiny AES in C» github, 2018.

PROTECTION OF MESSAGES BETWEEN SERVER AND IOT DEVICES

© 2019 I. E. Pustylnick, Yu. P. Preobrazhensky

Voronezh Institute of High Technologies (Voronezh, Russia)

In the Internet of things there is a variety of devices for which standard systems and algorithms od defense cannot be applied because of the small capacity of processors and performing any software operations on such devices. These devices require supporting software, which is based on the protocol of the application layer, named CoAP and the protocol of the physical layer, named ZigBee. Apart from this, such devices of the Internet of Things require coding/decoding protocols, which use the most modern algorithms of such defense, namely AES/SHA256 and a lightweight realization called TinyAES.

Keywords: AES, TinyAES, CoAP, ZigBee.