计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(10):178−182 [doi: 10.15888/j.cnki.csa.007124]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 MQTT 的数据加密传输算法①

于振中[1], 洪辉武[1], 徐　国[2], 朱丹青[2]

[1](江南大学 物联网工程学院, 无锡 214122)
[2](哈工大机器人 (合肥) 国际创新研究院, 合肥 230601)
通讯作者: 洪辉武, E-mail: honghwjn@163.com

摘　要: 提出了一种改进 MQTT 协议的数据传输加密算法 MQTT-EA (MQTT Encryption Algorithms). 该算法中, 物联网设备端与服务器端随机生成自己的私钥, 然后相互通知对方自己的私钥并通过算法组合成最终的会话主密钥, 通过 DES 加密、解密, 传输安全数据. 模拟了敌手 A、B 对数据传输过程进行攻击, 验证了在会话密钥生成算法没有泄露的前提下 MQTT-EA 是安全的.

关键词: MQTT 协议; 物联网; 数据加密传输; 无证书密钥协商; MQTT-EA

引用格式: 于振中,洪辉武,徐国,朱丹青.基于 MQTT 的数据加密传输算法.计算机系统应用,2019,28(10):178–182. http://www.c-s-a.org.cn/1003-3254/7124.html

Data Encryption Transmission Algorithm Based on MQTT

YU Zhen-Zhong[1], HONG Hui-Wu[1], XU Guo[2], ZHU Dan-Qing[2]

[1](Internet of Things Engineering,Jiangnan University, Wuxi 214122, China)
[2](HRG HRG Institute (Hefei) for International Innovation, Hefei 230601, China)

Abstract: An improved data transmission encryption algorithm MQTT-EA (MQTT Encryption Algorithms) is proposed. In this algorithm, the IoT device and server sides randomly generate their own private keys, then notify each other of their own private keys and combine them into the final session master key through the algorithm. The secure data is transmitted through DES encryption and decryption. The data transmission process is simulated by rivals A and B, and MQTT-EA is proved to be secure as long as the session key generation algorithm is not leaked.

Key words: MQTT; Internet of Things (IoT); data encryption transmission; certificateless key agreement; MQTT-EA

在设计工业机器人数据采集系统的时候, 我们发现在物联网场景下, 应用最多的数据传输协议便是 MQTT (Message Queuing Telemetry Transport, 消息队列遥测传输协议). MQTT 是由 IBM 发布的一种基于发布/订阅 (publish/subscribe) 模式的轻量级通讯协议, 由于规范简单, 非常适合用于低性能和网络带宽有限的物联网领域. MQTT 协议支持绝大部分平台以及编程语言[1], 这也使得它的开发非常方便. 但是 MQTT 协议有一个非常大的缺陷, 它的数据传输默认是不加密的, 数据的安全性得不到保障.

许多作者在物联网的数据安全传输方面已经做了大量的研究. 巫钟兴和李辉提出了一种软硬件结合的数据加密传输方案[2], 这种方案需要使用硬件加密卡, 大大增加了数据的传输成本. 刘文浩和许春香提出了一种无双线性对的无证书两方密钥协商方案[3], 该方案基于无证书密码体制, 规避了证书管理需要占用大量资源的缺点, 但仍需要第三方密钥生成中心 (KGC) 为通信双方生成部分私钥. Raza 等人提出了一种适合在

① 基金项目: 国家重点研发计划 (2018YFB1306100)
Foundation item: National Key Research and Development Program of China (2018YFB1306100)
收稿时间: 2019-03-19; 修改时间: 2019-04-17, 2019-04-26; 采用时间: 2019-04-29; csa 在线出版时间: 2019-10-15

物联网场景使用的轻量级安全传输协议 Lithe[4], 该协议集成了 DTLS (Datagram Transport Layer Security) 和 CoAP (Constrained Application Protocol), 使用 DTLS 作为身份验证机密通信的底层协议, 并且使用 6LoWPA 标准来降低传输过程的能耗. Lesjak 等人设计了一套基于硬件的 TLS 客户端系统来保证 MQTT 协议的安全性[5], 这种方法同样存在成本与证书管理的问题. Sainandan Bayya Venkata 和 Prabhakara Yellai 等人提出了一种新的轻量级传输方法 LWTM (Light Weight Transport Method)[6]. LWTM 算法分为控制平面和数据平面, 控制平面传输物联网设备与服务器之间交换密钥生成参数、加密数据块在消息中的位置、加密数据块的大小等信息, 使用安全传输通道 TLS 或 DTLS. 数据根据加密块的大小和加密块的位置进行加密, 并且通过 TCP/UDP 等不安全传输发送, 服务器通过使用相同的参数解密部分加密的数据. LWTM 通过数据部分加密的策略, 加快了加解密效率并且能够节省 CPU 计算资源. 高锐强, 朱虹的研究证明了通过 TLS/SSL 协议实现数据传输安全不可避免地会造成性能的损失[7].

1 MQTT-EA 算法

上文提到的方法存在成本高、资源消耗大、实现复杂等一些问题, 因此难以在我们的工业机器人数据采集系统中应用. 为了更好地满足数据采集系统的需要, 我们在 MQTT 协议的基础上, 自行设计了一种加密改进算法 MQTT-EA, 该算法有以下一些优点: (1) 该算法不需要额外的硬件支持、实现简单、成本更低; (2) 该算法是一种无证书协商算法, 不需要进行证书管理, 可以减少网络资源的浪费; (3) 该算法不需要依赖于任何第三方, 降低了私钥泄露的风险; (4) 该算法是基于 MQTT 协议而不是 CoAP 协议, 能够更好地支持多对多的通信; (5) 该算法是基于 MQTT 技术设计的, 由于 MQTT 技术已经非常成熟, 在各种编程语言中都有很好的支持, 在使用该算法进行项目开发的时候, 我们只要修改 mqttv3 jar 包以及 Apollo 服务器部份源码, 便能实现对 MQTT-EA 的支持, 这样能大大减少项目的工作量, 减少开发成本.

1.1 设计需求分析

作为物联网中使用最广泛的通信协议, MQTT 本身并没有提供数据加密的功能. 但大多数的物联网场景都对数据的安全性有很高的要求. 直接使用 MQTT 协议进行数据传输具有以下风险:

(1) 窃听风险 (eavesdropping): 第三方可以通过监听的手段轻易获知通信内容.

(2) 篡改风险 (tampering): 第三方可以修改通信内容.

(3) 冒充风险 (pretending): 第三方可以冒充他人身份参与通信.

为了弥补 MQTT 协议安全性的不足, 保障物联网设备的数据安全, 我们的改进算法必须具备以下特性:

(1) 所有数据都进行加密传输, 防止第三方进行数据窃取.

(2) 具有校验机制, 接收端能够校验数据是否被篡改.

(3) 具有身份验证机制, 防止第三方冒充我方设备.

1.2 安全策略

针对上文提出的 MQTT 协议的安全风险, MQTT-EA 在数据传输中采用了以下的安全策略:

(1) 黑白名单策略: 只有白名单中的设备才能够与服务器建立连接, 黑名单中的设备禁止与服务器连接.

(2) 格式验证策略: 设备端的数据必须按规定的格式存储, 如{rpm: 1000, temperature: 59, angle: 30}. SERVER 端将接收到的加密数据解密后, 首先对数据进行格式验证, 只有格式验证通过的数据才会被保存.

(3) 将最终会话生成密钥算法写入我们的设备及服务器, 最终会话密钥在客户端和服务器直接生成, 不在会话过程中出现, 这样就能保证最终会话密钥不会被破解.

(4) 强制中断连接: 在任意阶段只要发送了不符合要求的控制报文或者是验证未通过, 连接将会被强制中断.

(5) 数据加密传输: 所有的数据均通过 DES 加密后进行传输, 这样即使数据包被他人截获也不会造成数据泄露.

以上的安全策略的目的主要有两点, 一是加密数据, 并保证解密密钥不被破解, 二是实现敌我设备的认证. 在这些安全策略中, 加密数据的安全依赖于最终会话密钥生成算法不被破解, 设备认证则是通过设备号+白名单+数据格式认证的组合认证策略来实现的.

1.3 加密传输过程

MQTT-EA 的密钥协商是在 MQTT 建立连接的同时进行的, 流程如图 1 所示, 具体步骤为:

Step 1. 由 CLIENT 端向 SERVER 端发送一个 CONNECT+deviceID 控制报文. SERVER 端在接收到

报文后会对其中的 deviceID 进行验证, 只有在白名单中的设备才能进行下一步操作. 在 CLIENT 端与 SERVER 端成功连接后, SERVER 随机生成 ServerKey.

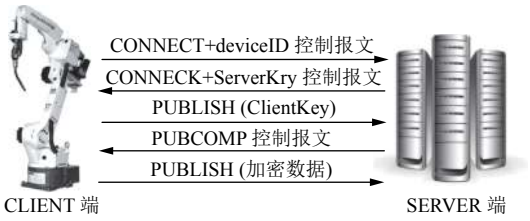

图 1 MQTT-EA 流程图

Step 2. SERVER端向CLIENT端发送一个CONNECK+ ServerKey 控制报文, CLIENT 端在接收到报文后将 ServerKey 提取出来存储到本地, 同时随机生成 ClientKey.

Step 3. 在 CLIENT 第一次进行 PUBLISH 时, 将自己的 ClientKey 发送到 SERVER 端. SERVER 端将已知的 ServerKey 和 ClientKey 作为变量, 利用自定义的秘钥生成算法 f (ServerKey, ClientKey) 获得会话主密钥 SessionKey.

Step 4. SERVER端向CLIENT端发送一个 PUBCOMP 控制报文, 表示数据传输前的密钥生成工作已经完成. CLIENT 端在接收到该报文后通过相同的密钥生成算法 f (ServerKey, ClientKey) 获得会话主密钥 SessionKey. 至此, CLIENT 端与 SERVER 端的连接成功建立, 可以进行数据的传输.

Step 5. CLIENT 端使用 SessionKey 对固定格式的数据进行 DES 加密, 然后再将加密后的数据 PUBLISH 到 SERVER 端.

Step 6. SERVER 端通过相同的 SessionKey 对接收到的加密数据进行 DES 解密, 之后对解密后的数据进行格式验证, 验证通过则将解密后的数据写到云端或者进行本地存储. 否者将舍弃该数据并将 CLIENT 的 ID 列入黑名单.

之后的数据传输过程重复 Step 5 和 Step 6 即可. 若是由于网络故障或者验证未通过等原因导致连接中断, 则需要从 Step 1 重新开始连接.

2 源码修改

为了使 mqttv3 jar 包和 apache apollo 1.7.1 服务器支持 MQTT-EA 的开发, 需要对客户端 mqttv3 jar 包以及部署在服务器的 apache apollo 1.7.1 源码进行改写, 部分改写如下:

(1) 在源码 MqttConnect 类中加入 deviceID 变量. 相应地服务器源码的解析类里面也需要添加白名单列表和对 deviceID 变量的解析方法.

(2) 在建立连接的时候服务器需要产生一个随机的 ServerKey, 因此需要在服务器相关类中加入生成 ServerKey 的方法.

(3) 服务器向 Client 返回 CONNACK+ ServerKey 控制报文. 相应地需要对 mqttv3 jar 包里面的 CONNACK 报文接收方法进行改写, 使其能够接收新的 CONNACK+ ServerKey 控制报文.

(4) 需要在服务器配置好获取会话主密钥 Session Key 的算法函数 f (ClientKey, ServerKey).

源码修改完成便能进行项目的开发了 mqttv3 jar 包以及 apache apollo 1.7.1 服务器对 MQTT-EA 的支持, 能让项目的开发更加方便、高效.

3 安全性分析

在我们的数据采集系统中, 可能受到的攻击有两种: (1) 敌手伪装成我们的设备, 向服务器发送虚假数据. (2) 敌手监听我们设备与服务器的会话, 窃取我们的设备数据. 我们将模拟系统遭受到这两种攻击的过程,验证系统的安全性.

假设有敌手 A, A 能将自己的设备号 device-A 加到白名单, 并且能够监听 CLIENT 端和 SERVER 端通信的任意阶段, 但是 A 不知道最终密钥的生成算法. A 的目的是伪装成正常的设备向服务器发送虚假信息. A 的攻击过程如下:

(1) A 向 SERVER 端发送一个 CONNECT+device-A 控制报文, 设备验证通过, SERVER 生成 $(ServerKey)_A$.

(2) SERVER 端向 A 发送一个 CONNECK+ $(ServerKey)_A$ 控制报文, A 保存 $(ServerKey)_A$ 并生成 $(ClientKey)_A$.

(3) A 将 $(ClientKey)_A$PUBLISH 到 SERVER 端, SERVER 端生成 $(SessionKey)_A$.

(4) SERVER 端向 A 发送一个 PUBCOMP 控制报文. A 在接收到该报文后生成错误的会话密钥 $(SessionKey)_{WRONG}$.

(5) A 使用 $(SessionKey)_{WRONG}$ 对需要传输的数据 DES 加密得到 DES{DATA, $(SessionKey)_{WRONG}$}, 然后

再将 DES{DATA, $(SessionKey)_{WRONG}$} PUBLISH 到 SERVER 端.

(6) SERVER 端使用 $(SessionKey)_A$ 对 DES {DATA, $(SessionKey)_{WRONG}$}解密后得到 $(DATA)_{WRONG}$. SERVER 端对数据进行格式验证, 验证失败. SERVER 端舍弃该数据, 强制断开与 A 的连接, 并将设备号 device-A 加入黑名单.

敌手 A 的攻击失败.

假设有敌手 B, B 同样能够监听 CLIENT 端和 SERVER 端通信的任意阶段, 但是 B 不知道最终密钥的生成算法. B 的目的是通过监听正常设备 C 与 SERVER 的会话获取我们的设备数据. B 的攻击过程如下:

(1) B 通过监听会话过程中的 Step 2 截获 SERVER 端向 C 发送的 CONNECK+$(ServerKey)_C$ 控制报文, 并从中提取出 $(ServerKey)_C$.

(2) B 通过监听会话过程中的 Step 3 截获 C 向 SERVER 端发送的 PUBLISH 控制报文, 并从中提取出 $(ClientKey)_C$.

(3) 由于该次会话的 SessionKey 从来没有在会话中出现过, 且 B 不知道会话密钥的生成算法, 所以 B 只能获得错误的会话密钥 $(SessionKey)_{WRONG1}$.

(4) B通过监听会话过程中的Step 5 截获C向SERVER端发送的加密数据包 DES{DATA, $(SessionKey)_C$}.

(5) B 使用 $(SessionKey)_{WRONG1}$ 对加密数据包 DES {DATA, $(SessionKey)_C$}解密, 解密失败.

敌手 B 攻击失败.

由以上分析可知, 在假设敌手不知道我们的最终会话密钥生成算法的假设下, MQTT-EA 可以满足我们对数据传输安全性的需求.

4 传输性能提高

MQTT 协议基于传统的 Publisher-Broker-Subscriber 模式, 数据由发布者传递到订阅者需要代理服务器作为中转, 经过了两次传输过程, 不仅增大了被窃听的风险而且降低了数据的传输效率. 本文中设计的 MQTT-EA 不再基于传统的 Publisher-Broker- Subscriber 模式, 而使用了 Client-Server 模式. 在这种模式下, 客户端与服务器在成功建立连接后, 直接将加密数据发送到服务器, 只需一次传输过程便可以实现数据的传递. 理论上, 采用 MQTT-EA 算法加密后 MQTT 的传输效率应该优于原始 MQTT 协议.

我们设计了一个简单的实验以验证使用 MQTT-EA 算法改进后的 MQTT 协议的传输效率高于原始 MQTT 协议. 对象一: HRG 工业机器人 A 使用 MQTT 协议向代理服务器发布一定量数据包, 再由 HRG 机器人云平台 A 订阅这些数据包并存储. 对象二: HRG 工业机器人 B 使用 MQTT-EA 改进的 MQTT 协议直接向 HRG 机器人云平台 B 发送等量数据包, 云平台 B 接收后将数据包解密并存储. 分别对对象一和对象二多次实验记录下消耗的时间, 通过对比两者传输相同数据包的平均用时, 我们便能知道它们的效率高低. 实验参数如表 1 所示.

表 1 实验的一些关键参数

协议	MQTT 协议	MQTT-EA 改进的 MQTT 协议
服务器设备	HRG 工业机器人 A	HRG 工业机器人 B
终端	HRG 机器人云平台 A	HRG 机器人云平台 B
代理	apache apollo 1.7.1	无
数据加密	无	DES 加密
数据内容	温度、转速、角度	温度、转速、角度
数据包个数 (单位: 万)	1、2、3、4、5	1、2、3、4、5

表注: HRG 工业机器人 A 配置了 MQTT 客户端, 可以进行数据的发布, HRG 机器人云平台 A 配置有 MQTT 客户端, 可以进行数据的订阅. HRG 工业机器人 B 配置了 MQTT-EA 改进 MQTT 客户端, HRG 机器人云平台 B 配置有修改版 apache apollo 1.7.1. 除了上面介绍的不同, HRG 工业机器人 A、B 完全相同, HRG 机器人云平台 A、B 完全相同

实验结果如图 2 所示, 在硬件及网络条件相同的情况下, 使用 MQTT-EA 改进的 MQTT 数据传输相同数量的数据包所花费的时间小于 MQTT 协议, 这说明使用 MQTT-EA 改进 MQTT 协议在保证数据安全性的同时, 还提高了它的数据传输性能.

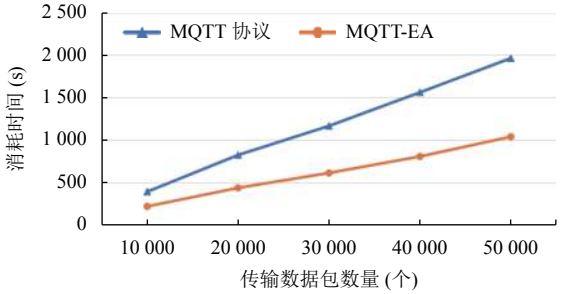

图 2 两种协议传输相同数量数据包耗时对比

5 总结与展望

本文根据工业机器人数据采集系统中数据传输的安全性需求, 在 MQTT 的基础上提出了 MQTT 的加密

算法 MQTT-EA. 该算法实现简单、传输效率高, 且具有不需要额外的硬件支持, 不需要证书管理, 不需要第三方 KGC 参与私钥生成, 开发效率高等一系列优点. 由此可见, MQTT-EA 算法具有较高的实用价值, 可以在物联网的数据传输中推广使用.

参考文献

1 许金喜, 张新有. Android 平台基于 MQTT 协议的推送机制. 计算机系统应用, 2015, 24(1): 185–190. [doi: 10.3969/j.issn.1003-3254.2015.01.035]

2 巫钟兴, 李辉. 一种数据加密传输方案的设计与实现. 北京化工大学学报 (自然科学版), 2011, 38(2): 104–107. [doi: 10.3969/j.issn.1671-4628.2011.02.021]

3 刘文浩, 许春香. 无证书两方密钥协商方案. 软件学报, 2011, 22(11): 2843–2852. [doi: 10.3724/SP.J.1001.2011.03942]

4 Raza S, Shafagh H, Hewage K, *et al*. Lithe: Lightweight secure CoAP for the internet of things. IEEE Sensors Journal, 2013, 13(10): 3711–3720. [doi: 10.1109/JSEN.2013.2277656]

5 Lesjak C, Hein D, Hofmann M, *et al*. Securing smart maintenance services: Hardware-security and TLS for MQTT. Proceedings of the 2015 IEEE 13th International Conference on Industrial Informatics. Cambridge, UK. 2015. 1243–1250.

6 Venkata SB, Yellai P, Verma GD, *et al*. A new Light Weight Transport Method for secured transmission of data for IoT. Proceedings of 2016 IEEE International Conference on Advanced Networks and Telecommunications System. Bangalore, India. 2016. 1–6.

7 高锐强, 朱虹, 贾立东, 等. 基于 SSL 安全协议实现工业控制通讯协议加密及认证的研究. 化工设计通讯, 2019, 45(1): 121–123. [doi: 10.3969/j.issn.1003-6490.2019.01.108]