

# Modbus TCP 安全协议的研究与设计

罗 旋 李永忠

(江苏科技大学计算机学院, 镇江, 212003)

**摘 要:** 针对 Modbus TCP 协议存在的安全问题, 如缺乏认证、数据明文传输和功能码滥用等, 基于原 Modbus TCP 协议提出一种安全的工控通信协议 (Modbus-S 协议): 采用数字签名技术保证数据的完整性和可认证性; 使用对称密钥技术保证数据的机密性; 利用哈希算法的单向性原理保证数据的唯一性; 最后使用“白名单”过滤机制基于角色对功能码进行管理, 保证指令的可控性。实验验证和分析表明: Modbus-S 协议可以全面弥补 Modbus TCP 协议的设计缺陷, 与已有方法相比, 该方法安全性更高, 可以全面提高 Modbus TCP 协议的通信安全。

**关键词:** Modbus TCP 协议; 哈希算法; 对称密钥; 白名单; 身份验证

**中图分类号:** TP393.04      **文献标志码:** A

## Research and Design of Security Protocol for Modbus TCP

Luo Xuan, Li Yongzhong

(School of Computer Science, Jiangsu University of Science and Technology, Zhenjiang, 212003, China)

**Abstract:** Aiming at the security problems of Modbus TCP protocol, such as lack of authentication, data transmission in clear text and abuse of function codes, a secure industrial control communication protocol (Modbus-S protocol) based on the original Modbus TCP protocol is proposed. The digital signature technology is used to ensure the integrity and authentication of data. The symmetric encryption is used to ensure the confidentiality of data. The unidirectional principle of hash function is used to guarantee the uniqueness of data. Finally, the "white list" filtering mechanism is used to manage function codes based on roles to ensure the controllability of instructions. Experimental verification and analysis show that Modbus-S protocol can fully compensate for the design defects of Modbus TCP protocol. Compared with the existing methods, the method has higher security and can comprehensively improve the communication security of Modbus TCP protocol.

**Key words:** Modbus TCP protocol; hash algorithm; symmetric key; white list; authentication

## 引 言

针对 Modbus TCP 协议存在的安全问题, 不少学者做了相关研究, 如高栋梁<sup>[1]</sup>以 netfilter 为框架, 设计了一个 Modbus TCP 协议网关防火墙; Shang 等<sup>[2]</sup>基于 Linux 平台, 使用 iptable 工具设计了一个工业防

防火墙。但是二者并未对传输数据做加密处理,不能保证数据的机密性,其次无法防止重放攻击。尚文利等<sup>[3]</sup>提出了一种基于统计分析的日志规则自学习算法,动态生成并更新规则;吕雪峰等<sup>[4]</sup>利用snort设计了基于协议缺陷和系统状态的检测规则,当测量值匹配规则时,触发告警。这二者提出的方法对规则精度的设定具有极高的要求,一旦精度设置过大或过小,都会严重影响检测效果。Fovino等<sup>[5]</sup>通过添加哈希链字段并作签名的方法实现了数据的完整性与可认证性,但仍存在功能码的滥用问题。

针对上述文献中存在的问题,本文基于功能码管理、数据传输以及可行性等问题对Modbus协议作了深入研究。以原Modbus协议为基础,设计了一种安全的工控系统通信协议——Modbus-S协议。该协议可以实现通信双方的双向认证,同时保证数据的完整性以及唯一性。该协议对关键数据字段进行加密,同时还可对功能码进行管理,从而全面提升工控通信的安全。

## 1 Modbus TCP协议

### 1.1 Modbus TCP协议介绍

Modbus TCP协议工作在TCP/IP模型的应用层,采用主从通讯模式,Modbus的主站可以与从站采用一对一或多对多方式进行通讯,其报文格式<sup>[6]</sup>如图1所示。

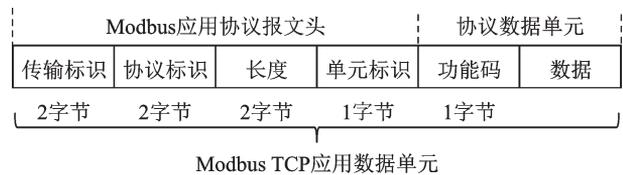


图1 Modbus TCP的报文格式

Fig. 1 Message format for Modbus TCP

### 1.2 Modbus TCP安全性分析

Modbus TCP协议由Modbus协议与TCP/IP融合而成,不仅未能解决原Modbus协议设计存在的安全隐患,甚至还引入了TCP/IP协议的安全缺陷<sup>[7]</sup>,如:

(1)缺乏认证机制<sup>[8]</sup>。任意客户端只要与服务器端网络互通,并可对服务器端发送指令。攻击者只需接入网络中,便可对服务器发送恶意指令。

(2)无权限管理<sup>[9]</sup>。所有用户均拥有相同权限,可执行任意指令,如误操作发送了重启或关机等高危命令,服务器端仍能正常响应。

(3)数据篡改。攻击者可以通过截获数据包,并篡改相应字段,从而实现网络攻击。

(4)数据明文传输。Modbus TCP协议以明文方式传输数据,一些敏感信息很容易被攻击者获取,攻击者可以通过收集相关信息为将来攻击系统作准备<sup>[10]</sup>。

(5)重放攻击。Modbus TCP缺乏防重放机制,攻击者可以反复发送截获的数据包,从而破坏系统的稳定性。

## 2 Modbus-S协议设计

### 2.1 Modbus-S协议格式

Modbus-S协议以原Modbus TCP协议为基础,通过添加相应字段,实现工控系统的通信安全。其协议格式如图2所示。

### 2.2 同步机制实现原理

通过利用哈希算法的单向性原理实现数据同步,其实现原理如下所示:

(1)首先取两个随机数依次作为种子 $a_{i,1}$ 和安全因子 $b_i$ ,每一对通信对象都拥有对应的种子与安全因子, $i$ 为其编号。 $V_{i,1}$ 为 $(a_{i,1} + b_i)$ 进行哈希计算的值, $F(x)$ 的值则为 $x$ 的首个字节数据,其作为同步标

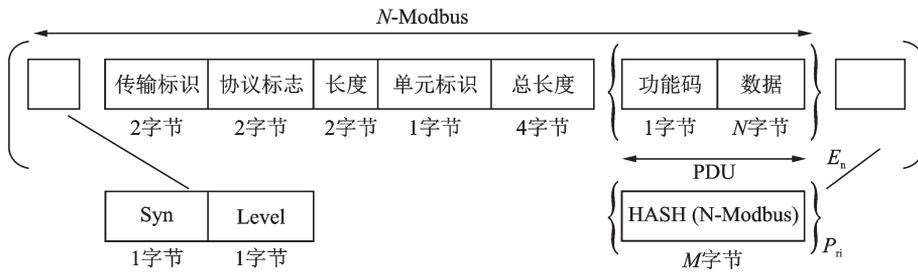


图2 Modbus-S协议格式  
Fig.2 Modbus-S protocol format

识字段  $Syn_{i,1}$  的值。

$$V_{i,1} = Hash(a_{i,1} + b_i) \quad i = 1, 2, \dots \quad (1)$$

$$Syn_{i,1} = F(V_{i,1}) \quad i = 1, 2, \dots \quad (2)$$

(2)若同步标识匹配成功后,则将当前哈希值与安全因子进行异或运算,然后求其哈希值,得到  $V_{i,2}$ ,同时取其首字节作为  $Syn_{i,2}$ 。

$$V_{i,2} = Hash(V_{i,1} + b_i) \quad i = 1, 2, \dots \quad (3)$$

$$Syn_{i,2} = F(V_{i,2} + b_i) \quad i = 1, 2, \dots \quad (4)$$

(3)为防止攻击者发送具有相同同步标识的数据包,破坏同步性,需要重复步骤(2)操作以确保每次在网络中传输的同步标识均不相同。得到  $V_{i,3}$  和  $Syn_{i,3}$ ,将  $Syn_{i,3}$  作为下次发送的同步标识码。

(4)使用相同方法得到每次通信的同步标识码。

通信双方通过匹配同步标识码,确定数据的唯一性。由于哈希算法的单向性,攻击者截获数据包,无法构造下一次发送的同步标识码。同时有安全因子的加入,以及数据包中不含有完整的哈希值,这也极大地提升了同步标识的安全性。

### 2.3 功能码管理机制

通过设定“白名单”过滤规则实现对功能码的有效管理,其规则模型<sup>[11]</sup>如图3所示。

Level	IP地址	设备ID	功能码
-------	------	------	-----

图3 “白名单”规则模型  
Fig.3 Rule model of "white list"

“白名单”过滤规则以 Level, IP 地址、设备 ID 以及功能码作为过滤对象。Level 字段的值用于进行角色划分,如 root, admin 和 monitor。root 权限最高,可执行任意指令;admin 的权限次于 root,只拥有线圈与寄存器值的读写权限;monitor 的权限最低,只可监视线圈或寄存器状态。多个用户也可以设置相同的权限等级,另外通过绑定 MAC 地址可以防止 ARP 攻击,提升网络的安全性。采用结构体数组方式存储“白名单”列表。每一条“白名单”规则对应结构体数组中的一个结构体,其参数由用户事先设定。

## 3 Modbus-S 协议系统的设计与实现

### 3.1 Modbus-S 协议系统实现原理

Modbus-S 协议系统通过自己开发的一对通信软件 Modbus-S client/server 实现数据的安全通信,以负责 Modbus TCP 与 Modbus-S 之间的协议转换,其实现原理如图4所示。

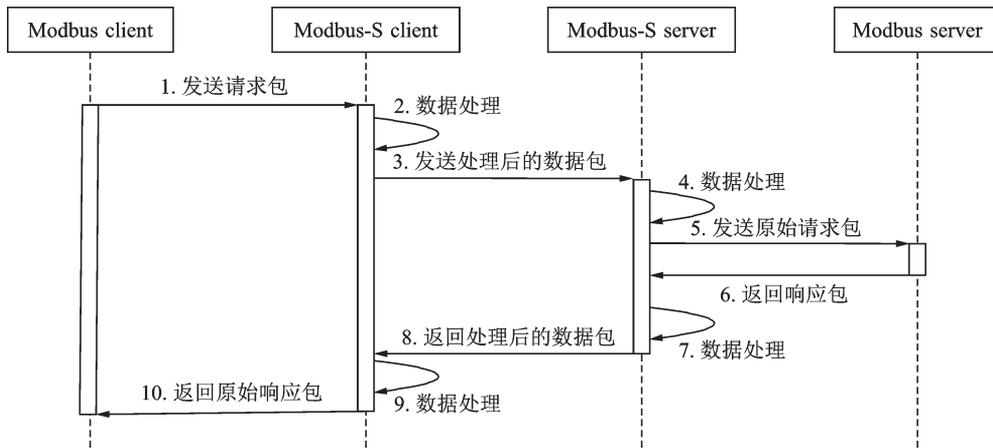


图4 Modbus-S协议系统实现原理

Fig. 4 Implementation principle of Modbus-S protocol system

### 3.2 Modbus-S协议数据处理过程

Modbus-S协议的数据处理由Modbus-S client与Modbus-S server实现,其处理过程为:

(1)Modbus client发送Modbus请求包( $M_{req}$ );

(2)Modbus-S client接收到 $M_{req}$ ,按照式(5,6)构造Modbus-S请求包 $C$ ,并发送给Modbus-S server端,其中各字段含义如下所示。

Syn:同步标识;Level:权限等级; $T_{ID}$ :传输标识; $P_{ID}$ :协议标识; $L$ :长度; $U_{ID}$ :单元标识; $L_1$ :加密后的PDU总长度; $E_n$ :对称密钥; $P_n$ :私钥。

$$N - \text{Modbus} = \text{Syn} | \text{Level} | T_{ID} | P_{ID} | L | U_{ID} | L_1 | \{PDU\} E_n \tag{5}$$

$$C = [ N - \text{Modbus} ] \{ \text{Hash}( N - \text{Modbus} ) \} P_n \tag{6}$$

(3)Modbus-S server接收到请求包 $C$ 后,首先计算本地的同步标识码,将其与请求包中的同步标识码进行匹配,若匹配成功则进行签名认证,其次再进行“白名单”粗过滤,即匹配Level,IP与设备ID字段的值。然后再对PDU进行解密,最后进行“白名单”细过滤,即匹配Level,IP,设备ID与功能码的值。返回其结果True或False。

$$R = \text{Check}(\text{Syn}_{sc}, \text{Syn}_{ss}) \& \text{Auth}(C) \& \text{WhiteListCheck}(C) \tag{7}$$

(4)只有当匹配结果为True时,才会构造原始请求包 $M_{req}$ 并转发,否则丢弃并告警。

(5)服务器端的响应包采用相同方法进行处理。

## 4 实验与分析

### 4.1 实验环境

实验以4台虚拟机与本地主机为环境,虚拟机与本地主机均为windows 7系统,使用Modbus poll/slave分别模拟Modbus TCP客户端与服务器端,使用自己开发的Modbus-S client/server分别模拟Modbus-S客户端与服务器端。虚拟机模拟Client端,本地主机模拟服务器端。其中3台虚拟机安装有Modbus poll与Modbus-S client。本地主机安装有Modbus slave与Modbus-S server。其中 $PC_1, PC_2, PC_3$ 均加入“白名单”列表,权限等级依次为100, 50, 10。其实验环境如图5所示。

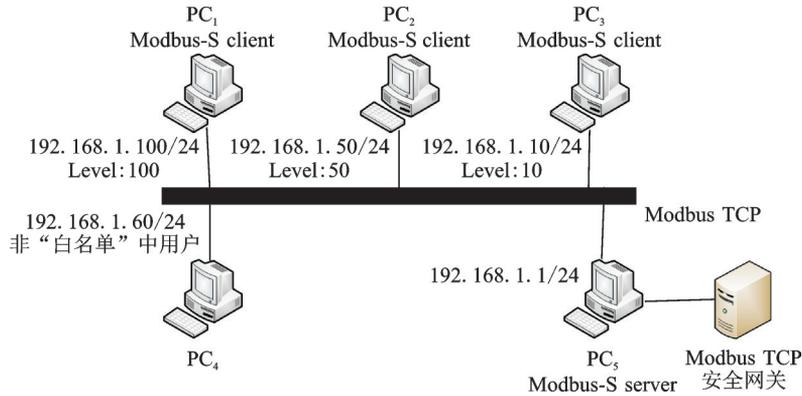


图5 实验环境图

Fig. 5 Experimental environment diagram

### 4.2 Modbus-S 窃听攻击测试

实验使用 Wireshark 抓取 Modbus-S 协议正常通信时的数据包进行分析。图 6 为 Modbus-S 请求包的 ADU 数据,图 7 为 Modbus-S 响应数据包 ADU 数据,由图可知数据的关键字段已被加密,测试结果符合预期。

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.100	192.168.1.1	TCP	1489 > 501 [SYN] Seq=0 Len=0 MSS=1460 WS=8
2	0.000114	192.168.1.1	192.168.1.100	TCP	501 > 1489 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
3	0.000280	192.168.1.100	192.168.1.1	TCP	1489 > 501 [ACK] Seq=1 Ack=1 Win=65536 Len=0
5	1.046720	192.168.1.1	192.168.1.100	TCP	501 > 1489 [PSH, ACK] Seq=1 Ack=138 Win=65536 Len=173
6	1.247203	192.168.1.100	192.168.1.1	TCP	1489 > 501 [ACK] Seq=158 Ack=174 Win=65280 Len=0
7	2.020232	192.168.1.100	192.168.1.1	TCP	1489 > 501 [PSH, ACK] Seq=158 Ack=174 Win=65280 Len=157
8	2.036703	192.168.1.1	192.168.1.100	TCP	501 > 1489 [PSH, ACK] Seq=174 Ack=315 Win=65536 Len=173
9	2.244318	192.168.1.100	192.168.1.1	TCP	1489 > 501 [ACK] Seq=315 Ack=347 Win=65280 Len=0
10	2.037816	192.168.1.100	192.168.1.1	TCP	1489 > 501 [PSH, ACK] Seq=315 Ack=347 Win=65280 Len=157

Hex	ASCII
0000 00 50 56 c0 00 08 00 0c 29 99 e4 b3 08 00 45 00	..P.V.....E.
0010 00 c3 44 ec 40 00 80 06 31 8f 60 a8 01 64 c0 a8	..D.B...1...d.
0020 01 01 05 01 f5 fe ef 86 19 81 f9 74 2e 50 18	...t...t.P.
0030 01 00 44 ec 00 00 7b 64 01 19 00 00 00 06 01 00	...D...d.....
0040 00 00 20 34 91 fd 60 ee d7 b3 10 db 88 9b 51 21	...P...d...g1
0050 cd 57 b8 f9 9c 09 2a db 1f ed 2c 57 f3 fb 33	...W...W...3
0060 44 57 00 07 50 45 63 15 c9 84 ff 08 80 42	...w...g...e...6
0070 72 c5 de 03 7e 47 de 73 0c 61 53 0a 8d 7a 34 1b	...c...G...u...a5...24.
0080 4e 06 e7 05 28 66 fa e0 d1 9a 02 85 e6 88 4a	...N...G...f...-...7
0090 0b 06 87 3b 17 fd f0 d0 f3 a2 0a 63 f0 02 9a	...b...o...f...-...7
00a0 94 09 04 80 7d 3c 0b c4 a8 39 44 75 14 f7 77 33	...-...<...9Du...u3
00b0 94 24 1f 34 2a 26 7a 60 90 97 ef 44 22 25 9c	...-...<...9Du...u3
00c0 ec 52 5f 02 30 68 f4 76 d6 a2 43 8a 27 9b ad 04	...R...Oh...V...c...1
00d0 3a 6e ad	...ln

图6 Modbus-S 请求数据包的 ADU 数据

Fig. 6 ADU data of Modbus-S request packet

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.100	192.168.1.1	TCP	1489 > 501 [SYN] Seq=0 Len=0 MSS=1460 WS=8
2	0.000114	192.168.1.1	192.168.1.100	TCP	501 > 1489 [SYN, ACK] Seq=0 Ack=1 Win=2097152 Len=0 MSS=1460 WS=8
3	0.000280	192.168.1.100	192.168.1.1	TCP	1489 > 501 [ACK] Seq=1 Ack=1 Win=65536 Len=0
4	1.012651	192.168.1.100	192.168.1.1	TCP	1489 > 501 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=157
5	1.016920	192.168.1.100	192.168.1.100	TCP	1489 > 1489 [ACK] Seq=1 Ack=138 Win=65536 Len=173
6	1.247203	192.168.1.100	192.168.1.1	TCP	1489 > 501 [ACK] Seq=158 Ack=174 Win=65280 Len=0
7	2.020232	192.168.1.100	192.168.1.1	TCP	1489 > 501 [PSH, ACK] Seq=158 Ack=174 Win=65280 Len=157
8	2.036703	192.168.1.1	192.168.1.100	TCP	501 > 1489 [PSH, ACK] Seq=174 Ack=315 Win=65536 Len=173
9	2.244318	192.168.1.100	192.168.1.1	TCP	1489 > 501 [ACK] Seq=315 Ack=347 Win=65280 Len=0
10	2.037816	192.168.1.100	192.168.1.1	TCP	1489 > 501 [PSH, ACK] Seq=315 Ack=347 Win=65280 Len=157

Hex	ASCII
0000 00 0c 29 99 e4 b3 00 50 56 c0 00 08 00 45 00	...P.V.....E.
0010 00 d5 77 40 40 00 80 06 ff 2c c9 a8 01 01 c9 a8	...w...e...d...o...o...
0020 01 04 01 f5 05 01 81 f9 74 2e fe ef 86 b6 50 18	...d...t...t...P.
0030 01 00 50 cf 00 00 7b 64 01 19 00 00 00 06 01 00	...D...d.....
0040 00 00 20 34 91 fd 60 ee d7 b3 10 db 88 9b 51 21	...P...d...g1
0050 3c 57 49 d4 87 18 09 03 0d 7b d2 06 ed 1b 22 5f	...d...i...t...e...f...5
0060 2a a8 70 c6 31 3f 3f 2a ee c5 45 09 3b 16 93	...P...t...e...c...e...5...9...b...16...93
0070 3b e2 10 13 95 a0 2d e2 e4 fc ab 62 89 f9 cf fe	...[P...A...N...E...C...-...b...-...7
0080 3b 35 35 6e 44 19 45 63 15 c9 84 ff 08 80 42	...c...c...-...b...-...7
0090 07 f9 63 b0 43 b2 a8 a8 3b a5 cf f5 09 83 c4 ee	...c...c...-...b...-...7
00a0 63 fc 09 82 6d c8 fd 92 30 62 f5 28 24 36 11 de	...[P...A...N...E...C...-...b...-...7
00b0 89 3b 2d 87 76 1c 05 09 53 81 c8 82 67 03 36	...[P...A...N...E...C...-...b...-...7
00c0 c9 20 3f ce b3 82 88 60 61 58 15 00 01 ff c3 51	...-...<...9Du...u3
00d0 89 40 f7 8f 50 45 0b 81 36 68 50 da 09 59 67 6c	...-...<...9Du...u3
00e0 8d d2 03	...-...<...9Du...u3

图7 Modbus-S 响应数据包的 ADU 数据

Fig. 7 ADU data of Modbus-S response packet

4.3 “白名单”过滤测试

表1为实验测试的“白名单”列表,表2为测试结果,数据通信数据包如图8所示,图9显示了告警日志。由实验结果可知“白名单”过滤方法可以有效地实现非法指令的过滤。

表1 实验测试“白名单”列表

Tab.1 "White list" of experimental test

Level	IP	Function	UnitID
100	192.168.1.100	All	All
50	192.168.1.50	1-6, F, 10H	All
10	192.168.1.10	1-4	All

表2 “白名单”实验测试结果

Tab.2 Results of "white list" test

PC	IP	FunctionID	UnitID	Result
PC <sub>1</sub>	192.168.1.100	3	9	Receive
PC <sub>1</sub>	192.168.1.100	6	9	Receive
PC <sub>4</sub>	192.168.1.60	3	6	Discard and alert
PC <sub>4</sub>	192.168.1.60	6	6	Discard and alert
PC <sub>2</sub>	192.168.1.50	3	5	Receive
PC <sub>2</sub>	192.168.1.50	6	5	Receive
PC <sub>3</sub>	192.168.1.10	3	1	Receive
PC <sub>3</sub>	192.168.1.10	6	1	Discard and alert

No.	Time	Source	Destination	Protocol	Info
1484	433.0583066	192.168.1.100	192.168.1.10	TCP	1196 > 501 [PSH, ACK] Seq=1 Ack=315 Win=64240 Len=157
1495	433.179453	192.168.1.1	192.168.1.100	TCP	501 > 1196 [PSH, ACK] Seq=1 Ack=158 Win=64240 Len=157
1496	433.379039	192.168.1.100	192.168.1.1	TCP	1196 > 501 [ACK] Seq=158 Ack=158 Win=64083 Len=0
1497	434.175419	192.168.1.100	192.168.1.1	TCP	1196 > 501 [PSH, ACK] Seq=158 Ack=158 Win=64083 Len=157
1498	434.199563	192.168.1.1	192.168.1.100	TCP	501 > 1196 [PSH, ACK] Seq=158 Ack=315 Win=64083 Len=157
1499	434.399750	192.168.1.100	192.168.1.1	TCP	1196 > 501 [ACK] Seq=315 Ack=315 Win=63926 Len=0
1500	436.339072	192.168.1.60	192.168.1.1	TCP	1173 > 501 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=12
1501	436.538852	192.168.1.1	192.168.1.60	TCP	501 > 1173 [ACK] Seq=1 Ack=13 Win=64240 Len=0
1502	436.078447	192.168.1.60	192.168.1.1	TCP	1173 > 501 [PSH, ACK] Seq=13 Ack=1 Win=64240 Len=12
1503	437.178520	192.168.1.1	192.168.1.60	TCP	501 > 1173 [ACK] Seq=1 Ack=25 Win=64228 Len=0
1511	439.078143	192.168.1.50	192.168.1.1	TCP	1169 > 501 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=157
1512	439.096724	192.168.1.1	192.168.1.50	TCP	501 > 1169 [PSH, ACK] Seq=1 Ack=158 Win=64240 Len=157
1513	439.296715	192.168.1.50	192.168.1.1	TCP	1169 > 501 [ACK] Seq=158 Ack=158 Win=64083 Len=0
1514	439.671248	192.168.1.50	192.168.1.1	TCP	1169 > 501 [PSH, ACK] Seq=158 Ack=158 Win=64083 Len=157
1515	439.688374	192.168.1.1	192.168.1.50	TCP	501 > 1169 [PSH, ACK] Seq=1 Ack=315 Win=64083 Len=157
1516	439.888779	192.168.1.50	192.168.1.1	TCP	1169 > 501 [ACK] Seq=315 Ack=315 Win=63926 Len=0
1520	442.435817	192.168.1.10	192.168.1.1	TCP	1204 > 501 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=157
1521	442.455443	192.168.1.1	192.168.1.10	TCP	501 > 1204 [PSH, ACK] Seq=1 Ack=158 Win=64240 Len=157
1522	442.669484	192.168.1.10	192.168.1.1	TCP	1204 > 501 [ACK] Seq=158 Ack=158 Win=64083 Len=0
1528	443.076142	192.168.1.10	192.168.1.1	TCP	1204 > 501 [PSH, ACK] Seq=158 Ack=158 Win=64083 Len=157
1532	443.276032	192.168.1.1	192.168.1.10	TCP	501 > 1204 [ACK] Seq=158 Ack=315 Win=64083 Len=0

图8 数据通信数据包

Fig.8 Data communication packet

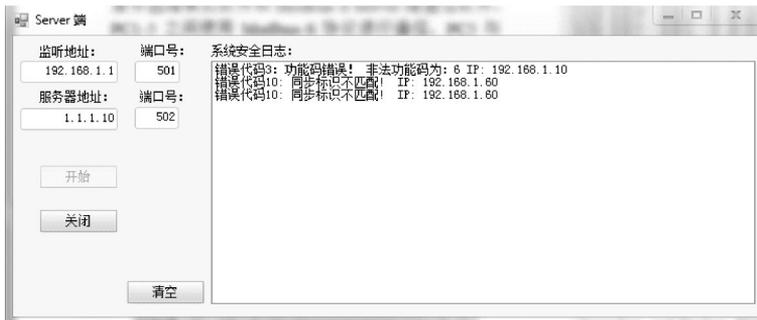


图9 Modbus-S 服务器端告警记录

Fig.9 Alarm record of Modbus-S server

#### 4.4 Modbus-S 时间性能分析

实验以使用最广泛的信息摘要算法 5(Message digest algorithm 5, MD5)、高级加密标准(Advanced encryption standard, AES)算法以及非对称加密(Rivest-Shamir-Adleman, RSA)数字签名算法进行分析。通过 wireshark 抓取通信链路的数据包,通过计算响应包与请求包的数据差值得到如图 10 所示的时间对比图。该图反映了 Modbus-S 协议进行一次通信所消耗的时间。通过对比图可知 Modbus-S 的数据包传输时间比 Modbus TCP 协议增加 8 ms 左右,比文献[5]增加 2 ms 左右。与提高安全性相比,这种差异可以忽略不计。

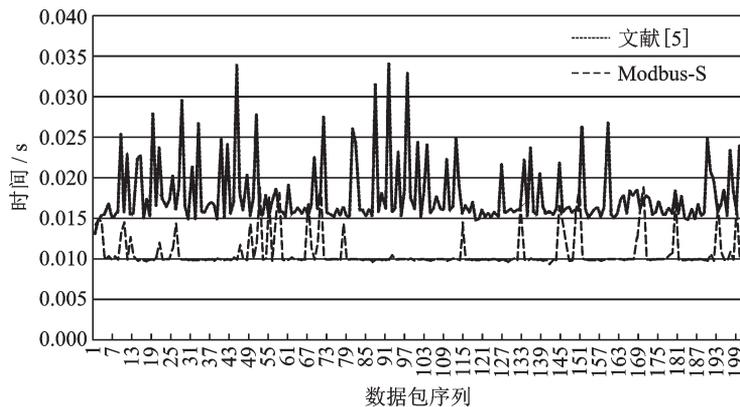


图 10 Modbus-S 数据包构造与解析时间对比图

Fig. 10 Time comparison chart of Modbus-S data packet construction and parsing

## 5 结束语

针对 Modbus TCP 协议存在的安全问题,本文通过开发一对通信软件并设计安全的通信协议 Modbus-S,实现了 Modbus TPC 协议的安全加固。由于原客户端通常为计算机,可直接部署相应软件处理数据,而服务器端通常为安全网关,需要借助其他终端部署对应软件。为降低实施成本,还需研究相应的硬件模块,这将是下一步需要研究的方向。

### 参考文献:

- [1] 高栋梁. MODBUS TCP/IP 协议防火墙的研究与实现[D]. 北京:北京邮电大学, 2015.  
Gao Dongliang. Research and implementation of MODBUS TCP/IP protocol firewall[D]. Beijing: Beijing University of Posts and Telecommunications, 2015.
- [2] Shang Wenli, Qiao Quansheng, Wan Ming, et al. Design and implementation of industrial firewall for modbus/TCP[J]. Journal of Computers, 2016, 11(5): 432-438.
- [3] 尚文利, 雷艳晴, 万明, 等. 基于哈希算法的工业防火墙规则自学习方法[J]. 计算机工程与设计, 2016, 37(12): 613-617.  
Shang Wenli, Lei Yanqing, Wan Ming, et al. Self-learning method for industrial firewall rules based on hash algorithm[J]. Computer Engineering and Design, 2016, 37(12): 613-617.
- [4] 吕雪峰, 蒋烈辉, 孟奕. 基于 MODBUS 的 SCADA 系统网络威胁与入侵检测[J]. 计算机工程与应用, 2017, 53(24): 122-128.  
Lv Xuefeng, Jiang Liehui, Meng Huan. Cyber threats and intrusion detection for MODBUS-based SCADA system[J]. Computer Engineering and Applications, 2017, 53(24): 122-128.
- [5] Fovino I N, Carcano A, Masera M, et al. Design and implementation of a secure modbus protocol[C]//International Conference on Critical Infrastructure Protection. Berlin, Heidelberg: Springer, 2009: 83-96.

- [6] Goldenberg N, Wool A. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems[J]. International Journal of Critical Infrastructure Protection, 2013, 6(2): 63-75.
- [7] 蔡宇晴. SCADA 系统中 Modbus TCP/IP 协议的异常检测研究[D]. 北京:北京交通大学, 2017.  
Cai Yuqing. Research on anomaly detection of Modbus TCP/IP protocol in SCADA systems[D]. Beijing:Beijing Jiaotong University, 2017.
- [8] 程超. 工业控制网络 Modbus TCP 协议深度包检测技术研究与应用[D]. 成都:电子科技大学, 2016.  
Cheng Chao. Research and implementation of deep packet inspection of Modbus TCP on industrial control network[D]. Chengdu: University of Electronic Science and Technology of China, 2016.
- [9] 万明, 尚文利, 曾鹏, 等. 基于功能码深度检测的 Modbus/TCP 通信访问控制方法[J]. 信息与控制, 2016, 45(2): 248-256.  
Wan Ming, Shang Wenli, Zeng Peng, et al. Modbus/TCP communication control method based on deep function code inspection[J]. Information and Control, 2016, 45(2): 248-256.
- [10] 胡爱群, 李古月. 无线通信物理层安全方法综述[J]. 数据采集与处理, 2014, 29(3): 341-350.  
Hu Aiqun, Li Guyue. Physical layer security in wireless communication: Survey[J]. Journal of Data Acquisition and Processing, 2014, 29(3): 341-350.
- [11] 张仁斌, 李思娴, 刘飞, 等. 基于 Modbus 功能码细粒度过滤算法的研究[J]. 计算机应用研究, 2018, 35(1): 277-281.  
Zhang Renbin, Li Sixian, Liu Fei, et al. Research on fine grained filtering algorithm based on Modbus function code[J]. Application Research of Computers, 2018, 35(1): 277-281.

## 作者简介:



罗旋(1993-),男,硕士研究生,研究方向:工控网络安全, E-mail: 851033149@qq.com。



李永忠(1961-),男,教授,硕士生导师,研究方向:计算机网络通信与安全、信息安全和嵌入式系统应用等。

(编辑:王静)