·网络空间安全· 文章编号：1000-3428(2021)08-0149-08 文献标志码：A 中图分类号：TP309

适用于PLC的时间基一次性密码方案

包致婷[1]，柯俊明[2]，杨 铮[1,3]，龙 华[1]，黄 东[4]
（1.重庆理工大学 计算机科学与工程学院，重庆 400054；2.山东大学 计算机科学与技术学院，济南 250000；
3.新加坡科技设计大学 信息系统技术与设计系，新加坡 138682；4.重庆机电职业技术大学 信息工程学院，重庆 402760）

摘 要：针对现有时间基一次性密码方案无法高效运行于可编程逻辑控制器(PLC)的问题，借鉴T/KEY单链方案，提出一种基于分组密码的时间基一次性密码方案BC-TOTP。使用PRESENT和SPECK分组密码算法来实例化加密函数，采用该加密函数计算链上的所有节点，使得证明方可在相应的时间内向验证方证明其身份。通过基于理想密码模型和分组密码IND-CPA的安全假设验证了BC-TOTP方案的安全性，并在罗克韦尔Allen-Bradley PLC上的测试结果表明，其能大幅减少计算时间，且单链使用周期将近1年。

关键词：时间基一次性密码；可编程逻辑控制器；分组密码；身份验证；结构化文本

开放科学（资源服务）标志码（OSID）：

中文引用格式：包致婷，柯俊明，杨铮，等.适用于PLC的时间基一次性密码方案[J].计算机工程，2021，47(8)：149-156.
英文引用格式：BAO Z T，KE J M，YANG Z，et al. Time-based one-time password scheme for PLC[J]. Computer Engineering，2021，47(8)：149-156.

Time-based One-Time Password Scheme for PLC

BAO Zhiting[1]，KE Junming[2]，YANG Zheng[1,3]，LONG Hua[1]，HUANG Dong[4]
（1.School of Information Science and Engineering，Chongqing University of Technology，Chongqing 400054，China；
2.School of Computer Science and Technology，Shandong University，Jinan 250000，China；
3.Department of Information Systems Technology and Design，Singapore University of Technology and Design，Singapore 138682；
4.Information Engineering Institute，Chongqing Vocational and Technical University of Mechatronics，Chongqing 402760，China）

【Abstract】To solve the problem that existing Time-based One-Time Password(TOTP) schemes cannot run efficiently on Programmable Logic Controller(PLC)，a TOTP scheme called BC-TOTP is proposed based on block cipher. The scheme employs block cipher algorithms，including PRESENT and SPECK，to instantiate the encryption function，which is used to compute each node in the chain，so the prover can authenticate to the verifier in time.The security of BC-TOTP is verified with a security assumption based on the ideal cipher model and IND-CPA of the block cipher. Then the proposed scheme is tested on a PLC of Rockwell Allen-Bradley. Test results show that the scheme can significantly reduce the computational time，and its single chain life cycle reaches almost one year.

【Key words】Time-based One-Time Password(TOTP)；Programmable Logic Controller(PLC)；block cipher；identity authentication；Structured Text(ST)
DOI：10. 19678/j. issn. 1000-3428. 0059091

0 概述

《中国制造2025》[1]提出要推进信息化和工业化的深度融合，但随着工业互联网的发展，工业控制系统逐渐从最初的封闭状态向开放状态转变，使暴露在互联网上的工控设备越来越多，从而对工控设备及系统安全构成了巨大威胁。工业互联网的核心部件是信息物理系统(Cyber-Physical Systems，CPS)[2]，但传统CPS本身没有安全措施，导致近年来CPS受到一些大型攻击[3-5]。CPS的核心是控制物理过程的可编程逻辑控制器(Programmable Logic Controller，PLC)[6]，因为PLC可以直接控制物理过程，所以成为攻击者攻击的主要目标。根据工业安全国家联合实验室[7]的调查显示，全球的工控设备接入工业互联网数量从2011年80亿台增长到2019年的174亿台。对工控设备遭受攻击的数量进行统计，在2017年上半年受攻击设备的比例为

基金项目：国家自然科学基金(61872051)。
作者简介：包致婷(1994—)，女，硕士研究生，主研方向为密码学；柯俊明，硕士；杨 铮，副教授；龙 华(通信作者)，副教授、讲师；黄 东，教授。
收稿日期：2020-07-29 **修回日期：**2020-09-01 **E-mail：**longhua@cqut.edu.cn

36.61%,在2017年下半年该比例增长至37.75%,并且这一增长趋势一直延续到2018年。

CPS使用防火墙[8]保证监控和数据采集(Supervisory Control And Data Acquisition,SCADA)系统和PLC之间的安全通信和访问控制,但防火墙并不能提出针对具体某个PLC的直接身份认证,而现有PLC没有任何身份相关的机密信息,因此监控系统不能确定被监控PLC身份的真实性。为高效实现PLC的身份认证,引入时间基一次性密码认证要素作为特定PLC的身份证明凭证,有效解决了现有工业控制系统中常见安全通信协议Modbus[9]、DNP3[10]及OPC-UA[11]的认证要素问题。目前,使用较广泛且高效的身份验证方案为时间基一次性密码方案[12]。时间基一次性密码的身份验证方案由LAMPORT[13]提出,该方案使用哈希链的具体实例,即利用哈希函数计算链上每个节点的密码。文献[14]提出S/KEY系统,并在系统中实现和验证了哈希链。文献[15]提出时间基一次性密码(Time-based One-Time Password,TOTP)认证算法,该算法控制了密码的有效期。文献[16]结合S/KEY和TOTP方案提出时间基一次性密码身份验证方案T/KEY,但目前该方案并没有在任何商业化PLC上进行测试和应用。

在IEC 61131-3规定的PLC 4种标准编程语言中,结构化文本(Structured Text,ST)语言[17]更接近计算机高级编程语言,因此适用于实现密码算法。但由于ST是一种类似Python的解释性语言,没有提供底层优化,并且许多商业PLC并没有提供密码算法实现需要的移位功能,因此基于ST实现的程序运行速度较慢。文献[18]提出一种轻量级哈希函数PHOTON算法,文献[19]提出一种SPONGENT哈希算法。因为英特尔的CPU集成了AES指令集,所以基于AES优化的PHOTON哈希算法在PC上运行效率较高,而在PLC上没有类似的优化指令,因此使用哈希函数的时间基一次性密码并不适用于PLC。文献[20]提出的PRESENT密码算法和文献[21]提出的SPECK密码算法通过比较原子操作个数得到:满足128位安全性的PHOTON主要做了10万个赋值、2万个加法和1.5万个异或,SPONGENT主要做了42万个赋值、3 000个加法和6 000个异或,PRESENT主要做了5 000个赋值和60个异或,SPECK主要做了3 000个赋值、80个加法和100个异或。通过实验测得在罗克韦尔Allen-Bradley的PLC上,赋值、加法和逻辑运算的时间分别为1.17 μs、1.51 μs和2.3 μs。由于赋值操作时间和其他开销近似,并且这些算法的赋值语句数量较多,因此以赋值时间为参考,PHOTON和SPONGENT分别约为SPECK的33和140倍,相比分组密码,进一步验证了哈希在PLC上运行效率较低。

为解决上述问题,本文构造一种适用于PLC的时间基一次性密码方案BC-TOTP,创新性地将分组密码作为构建块,充分利用分组密码计算链中的每个节点,将其前身作为加密密钥的输入、常量作为加密消息、链尾节点作为初始验证点、其他节点作为用于身份验证的一次性密码。

1 相关工作

由于静态密码[22]容易被猜测和遭受暴力攻击,因此目前研究集中于时间基一次性密码。S/KEY[14]使用哈希函数进行实例化,其一次性密码的有效期是不确定的,很容易被盗窃和滥用,并且S/KEY在链的每次迭代过程中都使用相同的哈希函数,使得它更容易被破坏。T/KEY[16]结合了S/KEY和TOTP的方法,不仅无需在服务器上存储密码,而且验证密码在短时间内有效,但是T/KEY使用的哈希函数在PLC上的效率较低。

目前,大量研究方案均允许验证方和BC-TOTP设备之间进行双向数字通信[23-24]。文献[25]研究了多种基于二维码的身份验证方案,在LBD-QR-PIN方案中移动设备生成密钥对并将公钥发送到验证方。随后,验证方会生成一个随机的128位质询,并使用证明方的公钥对其进行加密,然后将其发送至身份验证设备。身份验证设备将质询编码为二维码,然后用户可以使用其移动设备进行扫描。移动设备使用其存储的私钥解密挑战,计算挑战的简短6位哈希,并将其呈现给用户。用户在身份验证设备上输入此6位数字代码,并将其发送到验证方进行验证。

文献[26]针对当前工业控制系统的可靠性需求,提出一种全新的密码算法活性证明(Proof of Aliveness,PoA),并基于单向函数(One-Way Function,OWF)的时间基一次性密码(以下简称为OWF-TOTP)构建PoA的具体算法实现。相比基于OWF的Lamport一次一密方案[13],该方案采用的OWF-TOTP在标准模型下被证明是安全的。但是,JIN等[26]通过实验证明,基于哈希函数实例化的OWF-TOTP在Raspberry Pi平台上效率更高,而且该方案还未在真实商业化PLC上进行测试。

2 预备知识

2.1 符号定义

本文使用的符号定义如表1所示。

表1 符号定义

Table 1 Symbol definition

符号	描述
κ	安全参数
$\|$	字符串的连接
$\|\cdot\|$	值的位长
$\xleftarrow{s}$	均匀随机抽取元素
<<<, >>>	循环左移,循环右移
$[a]$	0~$(a-1)$的整数集合

2.2 分组密码

本节描述了一个分组密码方案BC。该方案由3种概率性多项式时间算法(BC.Gen, BC.Enc, BC.Dec)组成,具体步骤如下:

1)初始化算法 $k \xleftarrow{\$}$ BC.Gen(1^κ)。该算法输入安全参数 κ,输出加密和解密使用的密钥 k,其中密钥位长为 l_k。

2)分组加密算法 $c \leftarrow$ BC.Enc(k, m)。该算法输入位长为 l_k 的加密密钥 k 和位长为 l_m 的消息 m,输出位长为 l_c 的密文 c,其中 $l_k \leqslant l_m = l_c$。

3)解密算法 $m \leftarrow$ BC.Dec(k, c)。该算法输入解密密钥 k 和1条密文 c,输出1个明文 m。

本节定义了一个安全相关游戏 $\mathrm{Game}^{\mathrm{IND\text{-}CPA}}_{\mathrm{BC},\mathcal{A}}(\kappa)$ 来形式化定义选择明文攻击下的不可区分性(INDistinguishability Under Chosen-Plaintext Attack, IND-CPA)。给定安全参数、分组密码方案BC、挑战者 $\mathcal{B}$ 和攻击者 $\mathcal{A}$,游戏过程具体如下:

1)挑战者 $\mathcal{B}$ 开始运行初始化算法,并向攻击者 $\mathcal{A}$ 提供密钥 k。

2)攻击者 $\mathcal{A}$ 选择2个消息 m_0 和 m_1 发送给挑战者,其中 $|m_0| = |m_1|$。

3)挑战者随机选择 $b \xleftarrow{\$} \{0,1\}$,然后运行加密算法BC.Enc,得到密文 $c \leftarrow$ BC.Enc(k, m_b),将结果返回给攻击者 $\mathcal{A}$。

4)$\mathcal{A}$ 根据 c 输出对 b 的猜测 b'。

5)若 $b' = b$,Game输出1,则攻击者赢得游戏。

定义1(IND-CPA安全) 若概率多项式时间攻击者以不可忽略的优势攻破了上述安全模型,则BC方案是IND-CPA安全的,其中,攻击者 $\mathcal{A}$ 的优势概率为 $\left|\Pr[\mathrm{Game}^{\mathrm{IND\text{-}CPA}}_{\mathrm{BC},\mathcal{A}}(\kappa)=1]-1/2\right|$。

2.3 PRESENT和SPECK分组密码算法

2.3.1 PRESENT分组密码算法

PRESENT是SP网络[27]结构的超轻量级分组密码算法,支持64位的分组长度和80位、128位的密钥长度,它一共需要31轮加密,其中每轮的轮密钥 $K_i (1 \leqslant i \leqslant 31)$ 为64位,在31轮加密完成后,K_{32} 再运行一次轮密钥加得到密文。本文主要关注密钥长度为128位的PRESENT算法,算法步骤具体如下:

1)密钥生成算法 $k \xleftarrow{\$}$ PRESENT.Gen(1^κ)。该算法输入安全参数 κ,输出密钥 k。

2)分组加密算法 $c \leftarrow$ PRESENT.Enc(k, m)。该算法输入加密密钥 k 和位长为 l_m 的消息 m,输出长度为 l_c 的密文 c。该算法首先做密钥编排得到轮密钥,将密钥寄存器中存储的128位密钥记为 $K = k_{127}k_{126}\cdots k_0$,在加密算法第 i 轮使用的轮密钥是密钥寄存器 K 最左边的64位,即 $K_i = k_{127}k_{126}\cdots k_{64}$。然后执行以下步骤更新密钥寄存器 $K = k_{127}k_{126}\cdots k_0$:

(1)密钥寄存器循环左移61位:

$(k_{127}k_{126}\cdots k_0) = (k_{66}k_{65}\cdots k_{68}k_{67})$

(2)换字盒(SBox)代换:

$(k_{127}k_{126}k_{125}k_{124}) = \mathrm{SBox}(k_{127}k_{126}k_{125}k_{124})$

$(k_{123}k_{122}k_{121}k_{120}) = \mathrm{SBox}(k_{123}k_{122}k_{121}k_{120})$

(3)$k_{66}k_{65}k_{64}k_{63}k_{62}$ 与加密轮数做异或运算:

$(k_{66}k_{65}k_{64}k_{63}k_{62}) = (k_{66}k_{65}k_{64}k_{63}k_{62}) \oplus i$

通过以上步骤得到轮密钥 $K = (K_0 \| K_1 K_2 \cdots K_{T-1})$,其中 $T = 32$。然后,利用轮密钥和消息 m 执行以下步骤:

(1)轮密钥加AddRoundKey。对于 $z \in [64]$,该过程输入64位的状态 s 和轮密钥 K_i,然后得到更新的状态 s 通过计算 $s[z] := s[z] \oplus K_i[z]$,其中,初始状态为消息 m,$K_i[z]$ 为轮密钥 K_i 的第 z 个比特。

(2)换字盒代换SBoxLayer。该过程将64位的输入状态 s 分为16个4位的半字 $w_{15}w_{14}\cdots w_0$,即 $w[a] = s[4a+3]$,其中 $a \in [16]$,然后通过查找换字盒 $\mathrm{SBox}(w[a])$,最后输出更新后的状态值 s。

(3)换位盒置换PBoxLayer。该过程将状态 s 进行重排,即 s 的第 i 位移动到 $\mathrm{PBox}(s[i])$ 指定的位置。

当31轮循环完成后,加密算法使用 K_{32} 再次运行轮密钥加AddRoundKey得到密文 c。

3)解密算法 $m \leftarrow$ PRESENT.Dec(k, c)。该算法为加密算法的逆过程,输入密钥 k 和1条密文 c,输出1个明文 m。

2.3.2 SPECK分组密码算法

SPECK分组密码支持不同的分组长度和密钥长度,$l_e \in \{16, 24, 32, 48, 64\}$ 为字的位长,$l_m = 2l_e$ 为分组长度,加密密钥的位长为 $l_k = l_e l_w$,其中 $l_w \in \{2, 3, 4\}$。如果 $l_e = 16$,则位移常量 $\alpha := 7$、$\beta := 2$,在其他情况下 $\alpha := 8$、$\beta := 3$,算法步骤具体如下:

1)密钥生成算法 $k \xleftarrow{\$}$ SPECK.Gen(1^κ)。该算法输入安全参数 κ,输出密钥 k,其中 $k = (k_0, k'_0, \cdots, k'_{l_w-2})$。

2)分组加密算法 $c \leftarrow$ SPECK.Enc(k, m)。该算法输入加密密钥 k 和1条消息 $m = (m_0 \| m_1)$,输出1个密文 $c = (c_0 \| c_1)$,其中,$|m_0| = |m_1|$,$c = (c_0 \| c_1)$。该算法先做密钥排程,即使用 k 生成每轮加密需要的轮密钥,对于 $i \in [T-1]$,具体计算如下:

(1)$k'_{i+l_w-1} := (k_i + (k'_i >>> \alpha)) \oplus i$。

(2)$k_{i+1} := (k_i <<< \beta) \oplus k'_{i+l_w-1}$。

在得到轮密钥 $K = (k_0 \| k_1 k_2 \cdots k_{T-1})$ 后,每轮加密过程都需要执行 $\mathrm{SPECK.RF}_k(x, y) = (x', (y <<< \beta) \oplus x')$,其中 $x' = ((x >>> \alpha) + y) \oplus k$,即整个加密过程为 $\mathrm{SPECK.RF}_{k_{T-1}} \circ \mathrm{SPECK.RF}_{k_{T-2}} \circ \cdots \circ \mathrm{SPECK.RF}_{k_0}$,其中 $\circ$ 代表加密过程的结合,运行 T 次轮函数。第1轮的初始状态为 $x = m_1$、$y = m_0$,最后一轮得到 $c_1 = x$、$c_0 = y$。

3)解密算法 $m \leftarrow$ SPECK.Dec(k, c)。该算法输入密钥 k 和1条密文 $c = (c_0 \| c_1)$,输出1个明文 $m = (m_0 \| m_1)$,利用 k 得到轮密钥 $K' = (k_{T-1} \| k_{T-2} k_{T-1} \cdots k_0)$。解密算法利用轮函数的逆,每轮执行 $\mathrm{SPECK.RF}^{-1}_k(x, y) = ((x \oplus k) - y') <<< \alpha, y')$,其中 $y' = (x \oplus y) >>> \beta$。第1轮的初始状态为 $x = c_1$、$y = c_0$,最后一轮得到 $m_1 = x$、$m_0 = y$。

3 BC-TOTP方案

本节详细介绍BC-TOTP方案在罗克韦尔Allen Bradley的PLC上的运行过程。由于基于哈希的时间基一次性密码不适用于PLC，因此本文设计一个通用的基于分组密码的时间基一次性密码方案BC-TOTP。链上的每个节点（尾节点除外）都是一个证明，用于在相应时间内向验证方证明身份，如果验证方在容忍时间内收到至少一个有效证明，则证明方身份验证成功，这些节点将以从尾节点到头节点的方向被消耗。同时，本节进一步学习了如何通过PRESENT和SPECK分组密码算法实例化BC-TOTP，并且基于理想密码模型[28]和分组密码IND-CPA的安全假设，证明了BC-TOTP的安全性。

3.1 基于分组密码的通用型TOTP方案

BC-TOTP是一个链式结构，它采用类似于T/Key的一次性密码，其链的每个内部节点（尾节点除外）都是一个身份认证的密码。对于$1\leqslant i\leqslant N$，链上的第i个密码$x_i=\mathrm{BC.Enc}(x_{i-1},m)$都可由头节点$x_0=k$计算得到，用于证明方在相应的时间内向验证方证明其身份。证明方保存头节点x_0，便于生成每一个验证密码，尾节点x_N中存在验证方，用于验证证明方发送的密码的正确性。证明方可以使用x_0得到BC-TOTP链上任何一个一次性密码，并将该密码发送给验证方进行身份验证。验证方在验证时，第一次使用的验证点为尾节点x_N，验证点是动态变化的，上一次身份验证成功的密码将成为下一个验证点。在BC-TOTP方案中，设置如下重要参数：Δ_{TL}为单链的使用周期，n为一次性密码的长度，N为单链的密码节点数，离散时间槽$\{t_i\}$为时间的流逝，即$t_{i+1}-t_i=\Delta_{\mathrm{I}}$，$\Delta_{\mathrm{I}}$为每个密码的验证有效期（一般为30 s），$t_{\mathrm{tol}}$为身份验证容忍时间，$t_{\mathrm{ack}}$为验证方收到有效密码的最新时间。BC-TOTP步骤具体如下：

1）初始化算法$(\mathrm{st_{idc}},\mathrm{st_{idv}})\leftarrow\mathrm{Setup}(k,N,t_{\mathrm{start}},\Delta_{\mathrm{TL}},t_{\mathrm{tol}})$。该算法在初始化阶段证明方输入头节点$x_0=k$，使用分组加密函数BC.Enc初始化分组密码链，从头节点x_0开始到尾节点x_N结束，其中$N=\left\lfloor\Delta_{\mathrm{TL}}/\Delta_{\mathrm{I}}\right\rfloor$。对于$1\leqslant i\leqslant N$，第$i$个密码节点$x_i=\mathrm{BC.Enc}(x_{i-1},m)$，其中$m\in[N]$。若证明方将计算后的尾节点值$x_N$发送给验证方，验证方将初始验证点$\pi_{\mathrm{idc}}$设置为$\pi_{\mathrm{idc}}=x_N$，则证明方将保持状态$\mathrm{st_{idc}}=(k,t_{\mathrm{end}},\mathrm{BC.Enc})$，验证方的初始状态设置为$\mathrm{st_{idv}}=(\pi_{\mathrm{idc}},t_{\mathrm{ack}},\mathrm{BC.Enc})$。

2）密码生成算法$x_t\leftarrow\mathrm{Prover}(\mathrm{st_{idc}},t)$。该算法的证明方输入当前时间$t$和状态$\mathrm{st_{idc}}=(k,t_{\mathrm{end}},\mathrm{BC.Enc})$，对于$i\in[M]$，证明方通过计算第$i$个密码$x_i=\mathrm{BC.Enc}(x_{i-1},m)$得到一次性密码$x_t$，其中$M=\left\lceil t_{\mathrm{end}}-t\right\rceil/\Delta_{\mathrm{I}}$，然后将$x_t$发送给验证方。

3）身份验证算法$s\leftarrow\mathrm{Verifier}(\mathrm{st_{idv}},x_t,t,t_{\mathrm{ack}})$。该算法的验证方首先检查密码接收时间$t$和最近一次身份验证时间$t_{\mathrm{ack}}$的差值是否小于容忍时间$t_{\mathrm{tol}}$，即$t-t_{\mathrm{ack}}<t_{\mathrm{tol}}$；然后检查当前验证点$\pi_{\mathrm{idc}}$是否可以由密码$x_t$计算得到。具体为：对于$i\in[Z],Z=\left\lceil t-t_{\mathrm{ack}}\right\rceil/\Delta_{\mathrm{I}}$，通过计算第$i$个密码$x_i=\mathrm{BC.Enc}(x_{i-1},m)$得到一次性密码$x_Z$，其中$x_0=x_t$。如果$x_Z$与$\mathrm{st_{idv}}$的验证点$\pi_{\mathrm{idc}}$相等，则证明方身份验证成功，输出$s=1$，同时验证方更新状态值$\mathrm{st_{idv}}$为$\pi_{\mathrm{idc}}=x_t$，$t_{\mathrm{ack}}=t$；否则身份验证失败，输出$s=0$。

如图1所示，最近验证方成功验证的密码为$\pi_{\mathrm{idc}}=x_{N-2}$，它将作为验证点检查下一个密码。证明方在时间$t=t_{\mathrm{end}}-\Delta_{\mathrm{I}}$时发送一次性密码$x_t=x_1$给验证方，验证方首先判断是否在容忍时间内收到一次性密码x_t，如果收到则运行$N-3$次分组加密函数BC.Enc得到结果x_Z，最终判断x_Z是否与验证点π_{idc}相等。如果相等，则身份验证成功，输出$s=1$，并且更新验证方的状态值$\mathrm{st_{idv}}$；否则身份验证失败，输出$s=0$。在图1中密码$x_t=x_1$到验证点π_{idc}的计算过程为$\mathrm{BC.Enc}(x_1,m)\circ\mathrm{BC.Enc}(x_2,m)\circ\cdots\circ\mathrm{BC.Enc}(x_{N-3},m)$，即共运行$N-3$次加密函数BC.Enc。

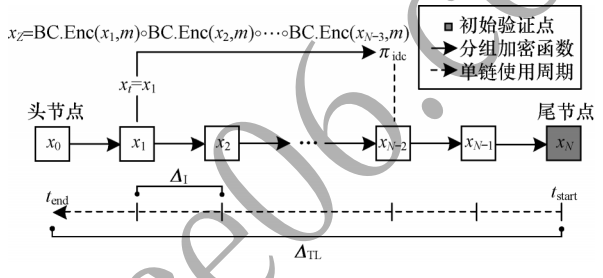

图1 基于分组密码的TOTP流程

Fig.1 Procedure of TOTP based on block cipher

3.2 算法实例

在随机预言模型中，本文使用分组密码实例化BC.Enc分组加密函数，并且利用分组长度为64位、密钥长度为128位的分组密码SPECK和PRESENT在ECB模式下实现BC.Enc。ECB是将整个明文分成若干段相同的小段，然后对每一小段进行加密。因为在分组加密函数BC.Enc中，输出的密文将作为计算下一个密文的密钥，所以需使用ECB模式连接两次具体分组密码实例的密文，这样才可以得到128位的密文。本节定义函数<Enc>表示具体分组密码实例，其<Enc>$\in\{\mathrm{PRESENT.Enc},\mathrm{SPECK.Enc}\}$的输入位长为$l_k$的密钥$k$和位长为$l_m$的消息$m$，输出$l_c$的密文$c$，其中$2l_m=2l_c=l_k$，计算如下：

$$c=\mathrm{BC.Enc}(k,m)=<\mathrm{Enc}>(k,m_0)\|<\mathrm{Enc}>(k,m_1)$$

其中，$m=(m_0\|m_1)$并且$|m_0|=|m_1|=64$。

对于$1\leqslant i\leqslant N$，BC-TOTP链中的每个密码节点可由下式计算得到：

$$x_i=<\mathrm{Enc}>(x_{i-1},m_0)\|<\mathrm{Enc}>(x_{i-1},m_1)$$

其中：$x_0=k$。

3.3　安全性分析

定理1　假设分组密码BC在理想密码模型下是IND-CPA安全的,则本文提出的BC-TOTP方案也是安全的。

证明　假设攻击者$\mathcal{A}$能够攻破BC-TOTP方案,即注入一个合法的密码x_{i-1},那么就可以构建一个高效的算法$\mathcal{B}$来攻破分组密码BC的IND-CPA安全性。

在理想密码模型下,由于初始密钥x_0是随机选择的,因此此后基于其产生的每一个已知密码$x_i=\mathrm{BC.Enc}(x_{i-1},m)$也是随机的并且具有唯一性。因此,$\mathcal{B}$可以首先猜测$\mathcal{A}$成功输出的密文的索引值为$i-1$,如果$\mathcal{B}$猜测正确则继续,否则终止实验,其中$\mathcal{B}$猜中的概率为$1/N$。在猜测正确的情况下,$\mathcal{B}$通过询问分组密码的挑战者,获得一个挑战密文$c^*=\mathrm{BC.Enc}(x_{i-1},m_b)$,其中消息$(m_0,m_1)$由$\mathcal{B}$任意选取。然后,$\mathcal{B}$基于密文$c^*$来模拟索引值$i$到$N$的所有密码。如果攻击者$\mathcal{A}$能成功输出密钥$x_{i-1}$,那么$\mathcal{B}$可以容易地通过对比$c^*$和基于密钥$x_{i-1}$与消息$(m_0,m_1)$分别产生的密文,从而返回正确的$b$来攻破BC的安全性。至此定理1得证。

4　算法实现与性能分析

4.1　PRESENT算法实现

PRESENT需要4位换字盒和64位换位盒,同时处理$l_m=64$的分组长度和支持两种长度$l_k\in\{80,128\}$的密钥,本文主要关注密钥长度为128位的密钥。首先将用数组$k[4]$表示的128位密钥k计算每轮64位的轮密钥,64位的轮密钥用$K[0],K[1]$表示,密钥的最后4位$(k_{127}k_{126}k_{125}k_{124})$可用$(k[3].[28],k[3].[29],k[3].[30],k[3].[31])$表示,因为PLC可以直接访问变量的每一位。然后执行轮密钥加、换字盒和换位盒的过程。最后该算法再运行一次轮密钥加得到$l_c=64$的密文c。

算法1　PRESENT分组密码算法

输入　消息数组$m[2]$、密钥数组$k[4]$

输出　密文数组$c[2]$

1.st[0]:=m[0]; st[1]:=m[1];
2.for i:=0 to T−1 by 1 do
3.K[0]:=k[2]; K[1]:=k[3];
4.密钥k左移61位;
5.tmp.[0]:=k[3].[28]; …; tmp.[3]:=k[3].[31];
6.tmp:=SBox(tmp);
7.tmp.[0]:=k[3].[28]; …; tmp.[3]:=k[3].[31];
8.tmp.[0]:=k[3].[24]; …; tmp.[3]:=k[3].[27];
9.tmp:=SBox(tmp);
10.k[3].[24]:=tmp.[0]; …; k[3].[27]:=tmp.[3];
11.K[1].[30],K[1].[31],K[2].[0],K[2].[1],K[2].[2]与i的有效低位做异或运算;
12.tmp:=0;
13.$\{st[i]\}_{i\in[2]}:=\{st[i]\}_{i\in[2]}\mathrm{XOR}\{K[i]\}_{i\in[2]}$;
14.$\{st[i]\}_{i\in[2]}:=\mathrm{SBox}(\{st[i]\}_{i\in[2]})$;
15.$\{st[i]\}_{i\in[2]}:=\mathrm{PBox}(\{st[i]\}_{i\in[2]})$;
16.end_for;
17.再次运行AddRoundKey得到最后的密文;
18.将密钥k清零

在算法1中:步骤1~步骤12是密钥编排过程,tmp获取密钥的其中4位;步骤13将内部状态与对应的轮密钥做异或运算;步骤14是SBoxPLayer通过查找换字盒每4位更新内部状态;步骤15、步骤16是PBoxPLayer实现的功能基于换字盒将内部状态的第i个比特即st.[i]移动到PBox[i]指定的新位置,这里表示为st[PBox[i]]:=st.[i];步骤17运行一次轮密钥加后得到最终的密文。由于考虑网络攻击者对PLC的读写攻击,因此在步骤18时进行密钥清零。图2给出了PRESENT密钥编排过程,并采用ST语言直接获取密钥寄存器所需的位。

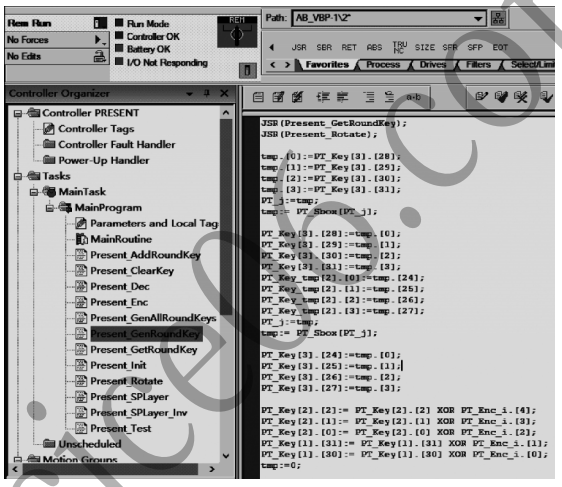

图2　PRESENT代码示例

Fig.2　Code example of PRESENT

4.2　SPECK算法实现

SPECK主要由循环移位、异或和加法运算组成。SPECK首先使用$(k[0],k'[0],k'[1],k'[2])$表示输入的128位密钥,计算加密算法所需的轮密钥$K[T]$,而非直接采用预计算的轮密钥。然后将输入分成$l_e=32$的两个消息,计算T次轮函数。最后得到$c[2]$表示的64位密文。

算法2　SPECK分组密码算法

输入　消息数组$m[2]$、密钥数组$k[T]$和$k'[T]$

输出　密文数组$c[2]$

1.for i:=0 to T−2 by 1 do
2.idx:=i+l_w−1;
3.k′[idx]:=(k[i]+k′[i]>>> α)) XOR i;
4.k[i+1]:=(k[i]<<< β) XOR k′[idx];
5.end_for;
6.c[0]:=m[0]; c[1]:=m[1]; α:=8; β:=3;
7.if l_e=16 then

8.α:= 7; β:= 2;
9.end_if;
10.for i:= 0 to T−1 by 1 do
11.c[1]:= ((c[1] >>> α) + c[0]) XOR k[i];
12.c[0]:= (c[0] <<< β) XOR c[1];
13.end_for;
14.将密钥k清零

在算法2中:步骤1~步骤5是轮密钥计算过程;步骤6~步骤9是判断α和β的值;步骤10~步骤13是算法加密过程;步骤14是将密文的密钥k清零。图3给出了SPECK密钥编排中的循环左移操作,该操作基于ST语言直接对变量的每一位进行赋值。

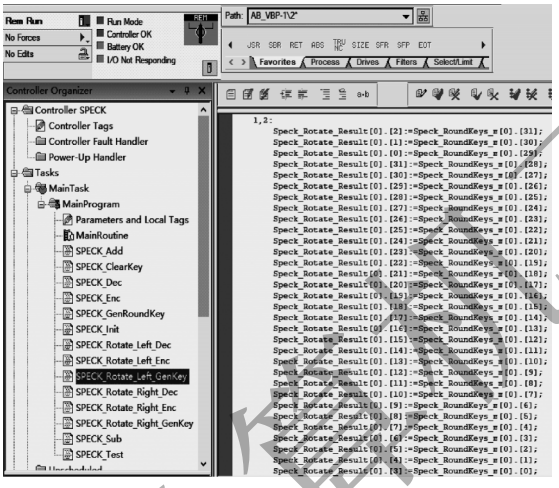

图3 SPECK代码示例

Fig.3 Code example of SPECK

4.3 BC-TOTP算法实现

BC-TOTP是利用分组密码实现的时间基一次性密码算法,数组k[4]表示128位的密钥,证明方使用密钥作为链的头节点x_0[4]计算每一个节点,利用$x_u[4]=<\mathrm{Enc}>(x_{u-1}[4], m_0)\,||<\mathrm{Enc}>(x_{u-1}[4], m_1)$得到链上的每一个节点,尾节点$x_N$[4]发送给验证方保存。在时间$t$时证明方可发送密码$x_t$[4]给验证方进行身份验证,验证方如果在容忍时间内收到密码x_t[4],然后x_t[4]经过Z次分组加密函数计算后得到的结果若等于π_{idc}[4],则证明方身份验证成功,输出$s=1$,同时验证方更新st_{idv}为$\pi_{idc}[4]=x_t[4]$,$t_{ack}=t$;否则身份验证失败,输出$s=0$。

算法3 BC-TOTP算法

1.{ x[0, i] } $_{i\in[4]}$:= { k[i] } $_{i\in[4]}$;
2.for u:= 1 to N by 1 do
3.{ x[u, i] } $_{i\in[4]}$:= BC.Enc({ x[u−1, i] }$_{i\in[4]}$, u);
4.end_for;
5.{ π_{idc}[i] } $_{i\in[4]}$:= { x[N, i] } $_{i\in[4]}$;
6.t_{ack}:= t_{start};
7.t_{end}:= t_{start} + Δ_{TL}; st_{idv} = (π_{idc}, t_{ack}, BC.Enc);
8.st_{idc}:= ({ k[i] } $_{i\in[4]}$, t_{end}, BC.Enc);
9.st_{idv}:= ({ π_{idc}[i] } $_{i\in[4]}$, t_{ack}, BC.Enc);
10.M:= (t_{end}−t) DIV Δ_1;
11.for u:= 1 to M by 1 do
12.{ x[u, i] } $_{i\in[4]}$:= BC.Enc({ x[u−1, i] }$_{i\in[4]}$, u);
13.end_for;
14.{ x[t, i] } $_{i\in[4]}$:= { x[M, i] } $_{i\in[4]}$;
15.if t−t_{ack} > t_{tol} then
16.身份验证失败;
17.end_if;
18.Z:=(t−t_{ack}) DIV Δ_1;
19.{ x′[0, i] } $_{i\in[4]}$:= { x[t, i] } $_{i\in[4]}$;
20.for u:= 1 to Z by 1 do
21.{ x′[u, i] } $_{i\in[4]}$:= BC.Enc({ x′[u−1, i] }$_{i\in[4]}$, u);
22.end_for;
23.if { x′[Z, i] } $_{i\in[4]}$:= { π_{idc}[i] } $_{i\in[4]}$ then
24.身份验证成功,输出s = 1,更新状态值
{ π_{idc}[i] } $_{i\in[4]}$:= { x′[0, i] } $_{i\in[4]}$; t_{ack} = t;
25.end_if

在算法3中:步骤1~步骤9是初始化过程,证明方计算尾节点x_N[4]将其发送给验证方,并且输出证明方和验证方的初始状态值st_{idc}和st_{idv};步骤10~步骤14证明方在当前时间t内计算一次性密码x_t[4]并发送给验证方;步骤15~步骤25验证方检查证明方提交密码的正确性,并更新状态值。

4.4 性能分析

本文在罗克韦尔Allen-Bradley的PLC上测试了BC-TOTP方案的性能。该PLC配有Controllogix 5571处理器和2 MB内存,利用1 000次重复实验的平均结果验证BC-TOTP方案的时间效率。分组长度64位,密钥长度128位的PRESENT分组密码算法的计算时间为14 ms,而同样版本的SPECK分组密码算法的计算时间为8.2 ms,并且它们都是在实现了密钥编排算法下的计算时间。因此,基于PRESENT和SPECK的链上相邻密码节点之间的计算时间分别为28 ms和16.4 ms。BC-TOTP方案的计算成本由密码数N决定,首先需要实例化这些参数,使得算法在PLC上可执行。如图4所示,若BC-TOTP方案的链上密码数N越大,则生成尾节点所需的计算时间越长。

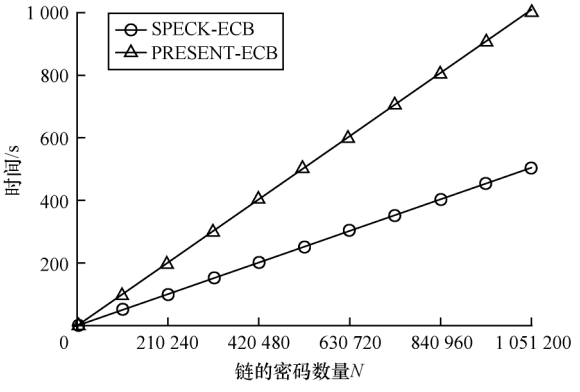

图4 SPECK和PRESENT算法的初始化时间比较

Fig.4 Comparison of initialization time between SPECK and PRESENT algorithms

由于BC-TOTP采用长链,为减少生成密码所需的加密次数,参考T/Key方案[16],根据证明方使用情况来缓存验证点。如图5所示,密码生成算法有worst和average两种情况,worst是在整条链上平均分布验证点,average验证点的位置选择是动态变化的。因为网络攻击者可以通过Pycomm读取和改写PLC上的变量值,以致PLC无法调整新的验证点,所以验证点必须预计算保存。图6给出了在worst情况下SPECK和PRESENT算法的身份验证时间比较结果。

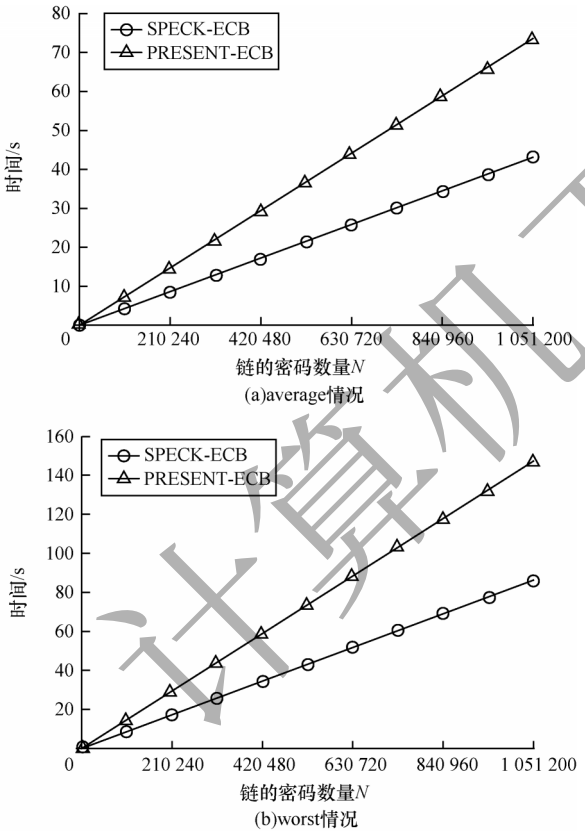

图5　SPECK和PRESENT算法的密码生成时间比较

Fig.5　Comparison of password generation time between SPECK and PRESENT algorithms

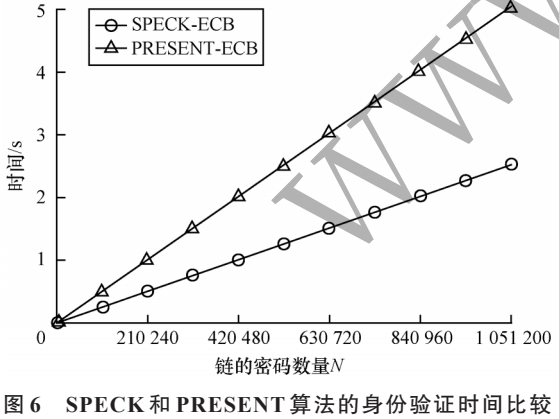

图6　SPECK和PRESENT算法的身份验证时间比较

Fig.6　Comparison of identity authentication time between SPECK and PRESENT algorithms

由图6看出,链上的节点越多,身份验证所需时间越长。本文考虑PLC内存大小,设置200个缓存检查点,并参考T/Key方案的密码使用间隔$\Delta_1=30$ s,如果$N=1\,051\,200$,则单链使用周期将近1年。

5　结束语

本文构造基于分组密码的时间基一次性密码方案BC-TOTP,利用PRESENT和SPECK分组密码算法对其进行实例化。通过基于理想密码模型和分组密码IND-CPA的安全假设证明了BC-TOTP方案的安全性,并在罗克韦尔Allen-Bradley PLC上的测试结果验证了BC-TOTP方案的高效性与实用性。后续将使用伪随机生成器链连接多个单链,打破单链密码数量的限制,构造一种全新的时间基一次性密码方案。

参考文献

[1] 王喜文. 中国制造2025解读[M]. 北京:机械工业出版社,2015.
WANG X W. Demystifying made in China 2025[M]. Beijing:China Machine Press,2015. (in Chinese)

[2] HUMAYED A, LIN J Q, LI F J, et al. Cyber-physical systems security—a survey[J]. IEEE Internet of Things Journal,2017,4(6):1802-1831.

[3] ZIELINSKA E, MAZURCZYK W, SZCZYPIORSKI K. Trends in steganography[J]. Communications of the ACM,2014,57(3):86-95.

[4] LIANG G Q, WELLER S R, ZHAO J H, et al. The 2015 Ukraine blackout: implications for false data injection attacks[J]. IEEE Transactions on Power Systems,2016,32(4):3317-3318.

[5] DI P A, DRAGONI Y, CARCANO A. TRITON: the first ICS cyber attack on safety instrument systems[EB/OL]. [2020-06-20]. https://scadahacker.com/library/Documents/Cyber_Events/Nozomi%20-%20TRITON%20-%20The%20First%20SIS%20Cyberattack.pdf.

[6] AHMED I, OBERMEIER S, SUDHAKARAN S, et al. Programmable logic controller forensics[J]. IEEE Security & Privacy,2017,15(6):18-24.

[7] 360 Internet Security Center. IT/OT integrated industrial information security situation report (2018)[EB/OL]. [2020-06-20]. https://zt.360.cn/1101061855.php?dtid=1101062514&did=610131448.

[8] PANATHUNGA D, ROUGHAN M, NGUYEN H, et al. Case studies of SCADA firewall configurations and the implications for best practices[J]. IEEE Transactions on Network and Service Management,2016,13(4):871-884.

[9] DUTERTRE B. Formal modeling and analysis of the Modbus protocol[C]//Proceedings of 2007 International Conference on Critical Infrastructure Protection. Berlin, Germany:Springer,2007:189-204.

[10] SIDDAVATAM I A, KAZI F. Security assessment framework for cyber physical systems: a case-study of DNP3 protocol[C]//Proceedings of 2016 Bombay Section Symposium. Washington D. C., USA: IEEE Press, 2016: 1-10.

[11] LEITNER S H, MAHNKE W. OPC UA—service-oriented architecture for industrial applications[J]. Softwaretechnik-Trends, 2006, 26(4): 61-66.

[12] 徐成强,李作维. 改进的一次性口令认证方案[J]. 计算机工程, 2009, 35(24): 168-169.
XU C Q, LI Z W. Improved scheme of one-time-password authentication[J]. Computer Engineering, 2009, 35(24): 168-169. (in Chinese)

[13] LAMPORT L. Password authentication with insecure communication[J]. Communications of the ACM, 1981, 24(11): 770-772.

[14] HALLER N. The S/KEY one-time password system: RFC 1760[S]. San Diego, USA: [s. n.], 1995: 151-157.

[15] M'RAIHI D, MACHANI S, PEI M, et al. TOTP: time-based one-time password algorithm [EB/OL]. [2020-06-20]. https://tools. ietf. org/html/draft-mraihi-totp-timebased-08.

[16] KOGAN D, MANOHAR N, BONEH D. T/KEY: second-factor authentication from secure hash chains [C]// Proceedings of 2017 ACM Conference on Computer and Communications Security. New York: ACM Press, 2017: 983-999.

[17] DARVAS D, VINUELA E B, MAJZIK I. PLC code generation based on a formal specification language[C]// Proceedings of the 14th International Conference on Industrial Informatics. Washington D. C., USA: IEEE Press, 2016: 389-396.

[18] GUO J, PEYRIN T, POSCHMANN A. The PHOTON family of lightweight hash functions[C]//Proceedings of the 31st Annual Cryptology Conference. Berlin, Germany: Springer, 2011: 222-239.

[19] BOGDANOV A, KNEZEVIC M, LEANDER G, et al. SPONGENT: the design space of lightweight cryptographic hashing [J]. IEEE Transactions on Computers, 2012, 62(10): 2041-2053.

[20] BOGDANOV A, KNUDSEN L R, LEANDER G, et al. PRESENT: an ultra-lightweight block cipher [C]// Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems. Berlin, Germany: Springer, 2007: 450-466.

[21] BEAULIEU R, SHORS D, SMITH J, et al. The SIMON and SPECK families of lightweight block ciphers [C]// Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference. Washington D. C., USA: IEEE Press, 2013: 404-449.

[22] HERLEY C, VAN O P. A research agenda acknowledging the persistence of passwords[J]. IEEE Security & Privacy, 2011, 10(1): 28-36.

[23] CZESKIS A, DIETZ M, KOHNO T, et al. Strengthening user authentication through opportunistic cryptographic identity assertions[C]//Proceedings of ACM Conference on Computer and Communications Security. New York, USA: ACM Press, 2012: 404-414.

[24] CHEN J S, EISENBARTH S C. Two-factor authentication for type 2 immunity[J]. Immunity, 2018, 49(3): 381-383.

[25] SHIRBANIAN M, JARECKI S, SAXENA N, et al. Two-factor authentication resilient to server compromise using mix-bandwidth devices[EB/OL]. [2020-06-20]. http://dev. www. isocdev. org/sites/default/files/08_3_slides. pdf.

[26] JIN C L, YANG Z, VAN D M, et al. Proof of aliveness[C]// Proceedings of the 35th Annual Computer Security Applications Conference. New York, USA: ACM Press, 2019: 1-16.

[27] SIVABALAN M, TAVARES S E, PEPPARD L E. On the design of SP networks from an information theoretic point of view[C]//Proceedings of the 19th Annual International Cryptology Conference. Berlin, Germany: Springer, 1992: 260-279.

[28] CORON J S, PATARIN J, SEURIN Y. The random oracle model and the ideal cipher model are equivalent[C]// Proceedings of the 28th Annual International Cryptology Conference. Berlin, Germany: Springer, 2008: 1-20.

编辑 陆燕菲