

边缘计算环境下轻量级终端跨域认证协议

朱宏颖, 张新有, 邢焕来, 冯力

(西南交通大学计算机与人工智能学院, 四川 成都 610097)

摘要: 边缘计算由于低时延、高带宽、低成本等众多优点, 被广泛应用在各种智能应用场景中, 但也因其分布式、实时性和数据多源异构性等特点, 面临安全方面的诸多挑战。身份认证是终端接入网络的第一步, 也是边缘计算的第一道防线, 为了解决边缘计算环境下的安全问题, 在“云-边-端”三级网络认证架构基础上, 提出了一种适用于边缘计算环境下的终端跨域认证协议。该协议首先基于 SM9 算法实现终端与本地边缘节点间的接入认证, 并协商出会话密钥; 然后利用该密钥结合对称加密技术和 Hash 算法实现终端的跨域认证; 认证过程中采用假名机制, 保护终端用户的隐私安全, 终端只需一次注册, 便可在不同安全域之间随机漫游。通过 BAN 逻辑证明了协议的正确性, 并对协议的安全性进行分析。结果表明, 该协议可以抵抗物联网场景下的常见攻击, 同时具备单点登录、用户匿名等特点。最后从计算成本和通信成本两方面对跨域认证协议进行性能分析, 并与现有方案进行对比。实验结果显示, 该协议在计算成本和通信开销上优于其他方案, 满足资源受限的终端设备需求, 是一种轻量级安全的身份认证协议。

关键词: 边缘计算; 身份认证; 多信任域; 轻量级

中图分类号: TP309

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023055

Lightweight terminal cross-domain authentication protocol in edge computing environment

ZHU Hongying, ZHANG Xinyou, XING Huanlai, FENG Li

School of Computer and Artificial Intelligence, Southwest Jiaotong University, Chengdu 610097, China

Abstract: Edge computing has gained widespread usage in intelligent applications due to its benefits, including low latency, high bandwidth, and cost-effectiveness. However, it also faces many security challenges due to its distributed, real-time, multi-source and heterogeneous data characteristics. Identity authentication serves as the initial step for terminal to access to the network and acts as the first line of defense for edge computing. To address the security issues in the edge computing environment, a terminal cross-domain authentication protocol suitable for the edge computing environment was proposed based on the "cloud-edge-end" three-level network authentication architecture. Access authentication was implemented between terminals and local edge nodes based on the SM9 algorithm, and

收稿日期: 2023-02-15; 修回日期: 2023-07-11

通信作者: 张新有, xyzhang@swjtu.edu.cn

基金项目: 国家自然科学基金 (62172342)

Foundation Item: The National Natural Science Foundation of China (62172342)

引用格式: 朱宏颖, 张新有, 邢焕来, 等. 边缘计算环境下轻量级终端跨域认证协议[J]. 网络与信息安全学报, 2023, 9(4): 74-89.

Citation Format: ZHU H Y, ZHANG X Y, XING H L, et al. Lightweight terminal cross-domain authentication protocol in edge computing environment[J]. Chinese Journal of Network and Information Security, 2023, 9(4): 74-89.

session keys were negotiated. The secret key was combined with symmetric encryption technology and hash function to achieve cross-domain authentication for the terminal. The pseudonym mechanism was used in the authentication process to protect the privacy of end users. The terminal only needs to register once, and it can roam randomly between different security domains. BAN logic was used to prove the correctness of the protocol and analyze its security. The results show that this protocol is capable of resisting common attacks in IoT scenarios, and it features characteristics such as single sign-on and user anonymity. The performance of the cross-domain authentication protocol was evaluated based on computational and communication costs, and compared with existing schemes. The experimental results show that this protocol outperforms other schemes in terms of computational and communication costs, making it suitable for resource-constrained terminal devices. Overall, the proposed protocol offers lightweight and secure identity authentication within edge computing environments.

Keywords: edge computing, identity authentication, multiple trust domain, lightweight

0 引言

5G 与物联网技术的快速发展,加快了万物互联时代的到来^[1],以智慧医疗、智能交通、智慧农业为代表的典型智能应用场景与移动通信技术深度融合,使得机器人手术、无人驾驶、智能收割等应用得以实现^[2]。这些应用需要大量的终端设备接入网络,并且对于数据的传输时延、处理速率、网络带宽等性能提出了新的要求。传统的数据处理主要通过云计算的方式实现,数据的传输需要耗费一定的时间,无法及时有效处理数据,且海量数据的集中式处理会导致核心网络负载较重^[3],进一步增加网络时延,对于机器人手术、无人驾驶这样的场景,微小的时延都会带来很严重的后果;同时,在数据传输到云中心的过程中存在用户隐私泄露的风险。可见,传统的云计算已经无法满足上述智能应用场景对于数据高效处理的需求,为了解决这些问题,边缘计算应运而生。

边缘计算就是将云中心的计算能力迁移至网络边缘^[4],不再将数据上传到核心管理平台而是在本地直接处理,这种计算下沉的方式,可以减轻云中心的计算压力,有效缓解网络带宽,减少通信时延,提供更快数据处理能力。通常将网络边缘处具备一定计算能力的设备称为边缘计算节点,简称边缘节点。我国作为边缘计算的重要领跑者,在2016年就组织成立了“边缘计算产业联盟”,目的就是推动各方产业资源合作,促进国内边缘计算行业的建设^[5],让边缘计算融入更多应用场景中,5G商用的持续推进更是加快了对边缘计算的发展^[6]。

然而,边缘计算在带来众多便利的同时,面临着安全与隐私保护方面的诸多挑战^[7]。在边缘计算环境下,数据被存储在不同的边缘网络中,多个安全域共存,导致终端在不同安全域之间的跨域认证面临挑战^[8];边缘节点设备的安全防护体系并不完善,容易遭受中间人攻击,一旦设备被盗,用户数据的完整性和机密性将遭到破坏;虽然终端数据不再需要远距离传输,但由于边缘节点设备可以主动获取用户数据,所以依旧存在用户隐私泄露的风险^[9];加上边缘计算的分布式、实时性和数据的多源异构性等特点,使得传统的身份认证和隐私保护体系不再适用,安全问题成为边缘计算能否可持续性发展的重要因素。

身份认证是终端接入网络的第一步,也是边缘计算的第一道防线^[10],有效的身份认证机制可以在双方通信时确定身份的合法性,保证数据的完整性和机密性。然而,业界对于边缘计算环境下身份认证技术的研究还处于探索阶段,相关成果较少,且大多数针对单一域内的身份认证问题进行研究,很少关注不同安全域下的跨域认证问题。

因此,本文主要针对边缘计算环境下跨域认证技术进行研究,提出一种轻量级的跨域认证协议。相比传统方案,该协议跨域认证过程中没有复杂的操作,以较低的开销即可实现终端与不同安全域的边缘节点相互认证,并协商出会话密钥,可满足资源受限的终端设备需求。

1 相关工作

在边缘计算环境下,终端设备数量多、移动性强,多种安全域共存^[5],所以不仅要研究单一

域内的身份认证,更要考虑终端在不同安全域间的跨域认证,这里的安全域主要是由具备相同安全保护需求且相互信任的边缘节点组成的逻辑区域。对于边缘计算环境下的跨域认证问题,相关研究成果如下。

Wang 等^[11]设计了一种跨域认证置信矩阵,通过优化可靠性矩阵,相同级别的终端可以高效地进行跨域认证;Fan 等^[12]使用基于属性加密的 CP-ABE 算法设计了一种边缘计算模型,解决了不同安全域之间数据共享的问题;吴卫^[5]基于 ECDH 密钥协商算法,设计了一种边缘计算环境下的终端跨域认证协议,在实现认证的同时协商出会话密钥,为后续通信提供安全保障;为了解决佩戴监测设备的老人与不同区域的边缘服务器相互认证的问题,Chen 等^[13]利用椭圆曲线设计了一种基于位置的身份认证方案,一旦这些设备检测到异常情况,可迅速与最近的边缘服务器进行切换认证,并发出报警信号请求治疗。

区块链分布式、中心化的特点使其成为跨域认证领域的研究热点。Sun 等^[14]针对复杂的“云-边-端”应用场景,基于离散对数难题,通过区块链技术在边缘服务器之间计算联盟密钥,借助联盟域签名实现终端的跨域认证;Lyu 等^[15]利用区块链的共识算法实现了物联网终端与不同边缘服务器之间的跨域认证,并利用 BAN 逻辑对协议进行了正确性证明;张金花等^[16]将 PKI 体制和区块链技术相结合,把终端证书的哈希值存在区块链中,通过联盟链的共识机制实现终端在边缘计算环境下的跨域认证。区块链技术虽然一定程度上能提高跨域认证的效率^[17],但节点数据的透明性容易导致终端设备隐私泄露;若每次跨域请求都上报到区块链,认证的同时会带来巨大的访问开销^[18]。

目前已有的研究成果虽然可以为边缘计算环境下的终端提供一种跨域认证机制,但往往伴随着较高的计算成本或者通信开销,操作复杂,灵活性和扩展性较差,且大多数方案不支持用户匿名,无法很好地保护终端身份隐私。

本文在上述成果的基础上,考虑到终端计算资源有限的问题,设计了一种轻量安全的跨域认证协议。该协议包括两部分,分别是终端的接入认证和跨域认证,其中,接入认证基于国密 SM9

算法实现,它是由国家密码管理局颁发的一种基于身份标识的密码算法,无须申请证书,大大降低了管理成本,能耗小,速度快,已成为 ISO/IEC 国际标准^[19];而跨域认证主要借助密码学中的对称加密技术和 Hash 算法实现。

2 方案设计

在认证协议中主要有 3 种角色,分别是终端、本地边缘节点 (MS1) 和其他安全域的边缘节点 (MS2、MS3),整个认证过程由初始化阶段、注册阶段、认证阶段三部分组成。终端首先向 KGC (key generation center) 注册获取私钥 (d_u, e_u) 和匿名身份 SID_u , 然后使用 SM9 算法实现与本地边缘节点 MS1 的接入认证,并协商出会话密钥 K_{c1} ;当终端移动到新的安全域时,需要利用 K_{c1} 结合对称加密技术实现与外部边缘节点 MS2 的跨域认证,并协商出新的会话密钥 K_{c2} ;边缘节点 MS1 与 MS2 之间需要通过 CA (certificate authority) 为它们颁发的证书实现相互认证。

方案的总体架构如图 1 所示。

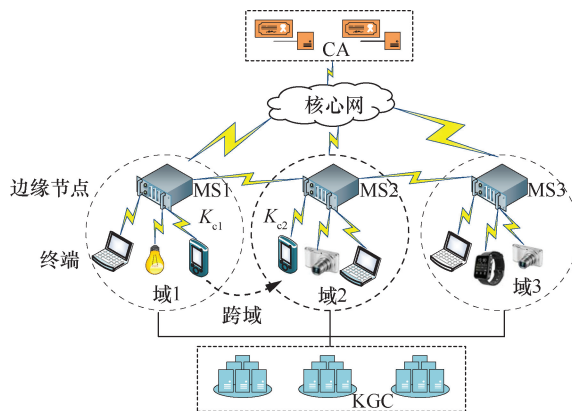


图1 方案的总体架构
Figure 1 Overall architecture of the scheme

2.1 初始化阶段

初始化阶段主要是 KGC 生成系统主密钥对的过程,因为终端的接入认证采用 SM9 算法,它基于 256 位 BN 椭圆曲线,并使用 Rate 双线性对,而曲线的参数是由 KGC 生成的,曲线方程为 $y^2 = x^3 + b$ 。KGC 是一个可信第三方,专门负责选择系统参数、为系统生成主密钥,它只初始化一次,为所有需要系统参数的设备提供服务。在初始化阶段主要是由 KGC 为系统生成 SM9 系统

参数, SM9 系统参数组如表 1 所示。

表 1 SM9 系统参数组
Table 1 SM9 system parameter group

符号	含义
cid	曲线识别符
F_q	椭圆曲线基域的参数
b	椭圆曲线方程参数
N	曲线阶的素因子
G_1, G_2	循环加法群
P_1, P_2	G_1, G_2 的生成元
e	双线性对
$(k_s, P_{\text{pub-s}})$	系统签名主密钥对
$(k_e, P_{\text{pub-e}})$	系统加密主密钥对

初始化阶段的主要过程如下。

1) KGC 选取随机数 $k_s \in [1, N-1]$, 作为系统的签名主私钥, 用于后续为终端生成签名私钥 d_u ; 利用 P_2 计算系统签名主公钥 $P_{\text{pub-s}} = k_s \cdot P_2$, 用于 SM9 数字签名算法。

2) KGC 选取随机数 $k_e \in [1, N-1]$, 作为系统的加密主私钥, 用于后续为终端生成加密私钥 e_u ; 利用 P_1 计算系统加密主公钥 $P_{\text{pub-e}} = k_e \cdot P_1$, 用于 SM9 的密钥封装算法。

3) KGC 选择并公开用一个字节表示的函数识别符 hid_s 和 hid_e , 分别用于 SM9 的数字签名、密钥封装算法。最后 KGC 秘密保存私钥 k_e 和 k_s , 公开 $P_{\text{pub-s}}$ 和 $P_{\text{pub-e}}$ 。

通过以上步骤, 完成系统参数组的初始化。

2.2 注册阶段

注册阶段主要是终端和边缘节点获取私钥的过程。终端的私钥由 KGC 计算得出, 包括签名私钥和加密私钥, 而边缘节点由于具备一定的计算能力, 私钥由自己计算产生。

2.2.1 终端注册过程

终端的注册是终端获取私钥的过程, 主要是由 KGC 根据终端身份标识, 利用 k_s 和 k_e 为其计算私钥, 并通过安全信道发给终端; 同时, 为了保护终端身份隐私, 还需要 KGC 利用 Hash 函数为终端计算假名, 此后终端都使用假名进行通信。终端注册过程的主要步骤如下。

1) 终端第一次注册时, 向 KGC 发送注册请求, 并发送自己的身份标识 ID_u , 为了保证私钥

的唯一性, 要求终端发送一个唯一的身份标识, KGC 收到请求后, 首先检查自己信任列表中是否存在该 ID_u , 若存在, 要求终端重新发送新的 ID_u , 否则, 由 KGC 为其计算相应的假名 SID_u 。

2) KGC 选择随机数 r_u , 利用单向散列函数计算匿名身份 $\text{SID}_u = H(\text{ID}_u \| r_u)$, 根据散列函数的单向性, 其他人无法反向解出 ID_u , 此后终端将使用 SID_u 进行匿名传输。

3) KGC 利用 SID_u 为终端计算私钥, 包括终端的签名私钥和加密私钥。在 F_N 上计算 $t_1 = H_1(\text{SID}_u \| \text{hid}_s, N) + k_s$, 若 t_1 等于 0, 则 KGC 需要重新选取系统参数中的签名主私钥 k_s , 并重新计算签名主公钥; 若 t_1 不为 0, 则计算 $t_2 = k_s / t_1$, 接着计算 $d_u = t_2 \cdot P_1 = k_s / (H_1(\text{SID}_u \| \text{hid}_s, N) + k_s) \cdot P_1$ 作为用户的签名私钥; 同理, KGC 计算用户的加密私钥 $e_u = k_e / (H_1(\text{SID}_u \| \text{hid}_e, N) + k_e) \cdot P_2$ 。

4) KGC 并不会保存终端的私钥, 而是把计算出的 d_u 、 e_u 和 SID_u 通过安全信道发给终端, 由终端秘密保存, 同时终端公开匿名身份 SID_u 。

综上所述, 完成终端的注册, 注册过程如图 2 所示。

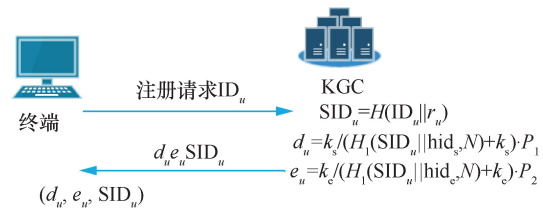


图 2 终端注册过程
Figure 2 Terminal registration process

2.2.2 边缘节点注册过程

边缘节点 (MS) 由于本身具备一定的计算能力, 所以由自己产生密钥对 $(P_{\text{ms}}, S_{\text{ms}})$, 其中, P_{ms} 是公钥, S_{ms} 是私钥, 边缘节点需要将公钥通过安全信道发给可信第三方 CA, 向 CA 申请数字证书 X。边缘节点注册过程的主要步骤如下。

1) 边缘节点利用 RSA 算法产生密钥对 $(P_{\text{ms}}, S_{\text{ms}})$, 然后将自己的身份信息与公钥 P_{ms} 一起发给 CA。

2) CA 收到信息后, 首先用自己的私钥对该信息进行签名, 然后把 CA 的签名、边缘节点的身份信息以及公钥 P_{ms} 放入数字证书, 除此之外, 证书还包括版本号、使用期限等信息; 最后 CA

将证书通过安全信道发给边缘节点。

3) 边缘节点收到证书后利用 CA 公钥进行验签, 以验证 CA 身份的真实性, 然后秘密保存证书, 边缘节点之间通过互相交换证书来确保对方公钥的真实性。边缘节点注册过程如图 3 所示。

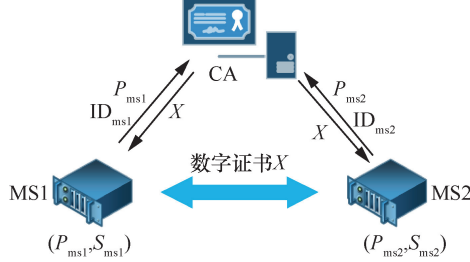


图 3 边缘节点注册过程
Figure 3 Edge node registration process

2.3 认证阶段

终端发起认证请求时, 首先会和距离最近的本地边缘节点 MS1 进行接入认证, 并协商出会话密钥 K_{c1} , 终端秘密保存 K_{c1} , 后续当终端移动位置进入其他安全域时, 可以利用 K_{c1} 和新的安全域中的边缘节点 MS2 进行跨域认证, 并协商新的会话密钥 K_{c2} 。认证过程中的相关符号及含义如表 2 所示。

表 2 认证过程中的相关符号及含义
Table 2 Symbols and meanings related to the certification process

符号	含义
MS1/MS2	不同安全域中的边缘节点
ID_{ms}	MS 身份标识
SID_u	终端匿名身份标识
(h, S)	终端的数字签名
KDF	密钥派生函数
K_{c1}	终端与 MS1 之间的会话密钥
$K_{cms}/K/K_{sm}$	随机数, 用作对称密钥
K_{c2}	终端与 MS2 之间的会话密钥
r_1, r_2, r_3	随机数
T_5, T_7	时间戳
$x y$	字符串 x 和 y 的拼接

2.3.1 终端的接入认证

在边缘计算环境下, 终端需要与本地边缘节点进行接入认证, 才能享受边缘节点设备提供的计算存储等服务。在本文协议中, 终端与边缘节点的接入认证是基于 SM9 算法的数字签名和密

钥封装算法实现的, 拟设计思路如下。

Step 1: 终端首先向本地距离最近的边缘节点 MS1 发送接入认证请求, MS1 收到请求后将自己的证书发给终端, 终端验证证书的合法性, 并从证书中取出边缘节点 MS1 的公钥 P_{ms1} 。

Step 2: 终端根据 SM9 算法的数字签名算法^[20], 利用系统参数、签名主公钥 P_{pub-s} 和终端的签名私钥 d_u 计算数字签名, 过程如下。

$$g = e(P_1, P_{pub-s}) \quad (1)$$

$$w = g^r \quad (2)$$

$$h = H_2(\text{Msg} || w, N) \quad (3)$$

$$S = (r - h) \bmod N \times d_u \quad (4)$$

生成的 (h, S) 就是数字签名, 其中, h 是一个数值, S 是一个点对的形式, 为了传输方便, 将生成的签名封装为 Signature 对象。终端选取一个随机数 K_{sm} , 作为与 MS1 之间的临时对称密钥, 最后终端用 P_{ms1} 将 Msg 和 K_{sm} 加密, 并附上匿名身份 SID_u 和 Signature 发给本地边缘节点 MS1。

$$C \rightarrow \text{MS1}: P_{ms1}(\text{Msg} || K_{sm}) || SID_u || \text{Signature} \quad (5)$$

Step 3: MS1 收到消息后, 利用 SM9 的验签算法进行验签, 过程如下。

$$g = e(P_1, P_{pub-s}) \quad (6)$$

$$t = g \times h' \quad (7)$$

$$h_1 = H_1(ID || \text{hid}, N) \quad (8)$$

$$P = h_1 \times P_2 + P_{pub-s} \quad (9)$$

$$w' = e(S', P) \times t \quad (10)$$

$$h_2 = H_2(\text{Msg} || w', N) \quad (11)$$

通过对比 h_2 与 h' 是否一致来完成验签, 若一致, 则验签成功, MS1 将终端的伪身份标识 SID_u 添加到自己的信任列表中, 否则直接视为认证失败。然后, 根据 SM9 的密钥封装算法进行密钥封装, 在进行封装时, 需要确定待封装的密钥长度 $klen$, 该长度由 MS1 确定并秘密保存, 然后利用系统加密主公钥 P_{pub-e} 和匿名身份标识 SID_u 按照以下公式进行密钥封装。

$$Q_D = [H_1(SID_u || \text{hid}_e, N)]P_1 + P_{pub-e} \quad (12)$$

$$C = r_1 \times Q_D \quad (13)$$

$$g = e(P_2, P_{pub-e}) \quad (14)$$

$$K_{c1} = \text{KDF}(C \parallel g^{r1} \parallel \text{ID}_{\text{ms}}, \text{klen}) \quad (15)$$

K_{c1} 即 MS1 封装出的会话密钥。MS1 秘密保存 K_{c1} ，并使用解密出来的 K_{sm} 将密文 C 、klen 和原签名信息 Msg 进行加密，同时附上自己的身份标识发给终端。

$$\text{MS1} \rightarrow C: K_{\text{sm}}(C \parallel \text{Msg} \parallel \text{klen}) \parallel \text{ID}_{\text{ms1}} \quad (16)$$

Step 4: 终端收到 MS1 的响应后，利用 K_{sm} 进行解密，得到 C 、 Msg 以及 klen ，终端首先对比接收到的 Msg 和之前发给 MS1 的是否一致，若不一致，则直接视为认证失败，否则按照 SM9 的密钥解封算法进行解封。

解封时首先判断 C 是否在椭圆曲线上，然后利用加密私钥 e_u 计算 $w' = e(C, e_u)$ ，最后计算会话密钥 $K'_{c1} = \text{KDF}(C \parallel w' \parallel \text{ID}_{\text{ms}}, \text{klen})$ ，若计算得出的密钥为 0，则密钥封装失败，需要告知 MS1 重新进行密钥封装；反之则封装成功，当 Msg 对比成功且密钥解封成功时，完成终端对 MS1 的身份认证。最后终端秘密保存 K'_{c1} ，并用 K'_{c1} 加密自己的匿名身份 SID_u 和边缘节点的身份标识 ID_{ms} ，向边缘节点发送认证成功反馈。

$$C \rightarrow \text{MS1}: K'_{c1}(\text{SID}_u \parallel \text{ID}_{\text{ms1}}) \quad (17)$$

Step 5: 边缘节点收到消息后利用 K'_{c1} 进行解密，理论上，根据 SM9 密钥封装算法，MS1 封装的 K'_{c1} 与终端解封的 K'_{c1} 应该是相同的，并且解封用到的 C 是通过加密发送的，只有对方可以知晓，所以若能成功解密，则说明对方成功实现了密钥解封，完成 MS1 对终端的身份认证。

终端的接入认证过程如图 4 所示。

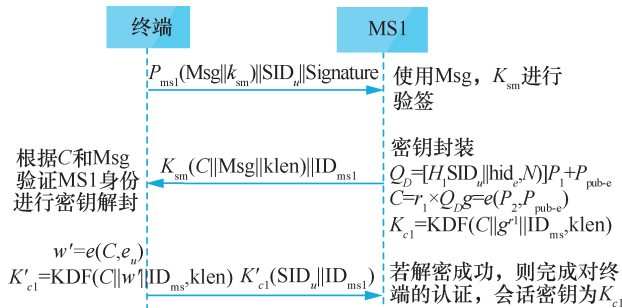


图 4 终端的接入认证过程

Figure 4 Access authentication process of terminal

2.3.2 终端的跨域认证

当终端设备移动到新的安全域时，需要与新

的安全域中的边缘节点 MS2 进行跨域认证，本文设计的跨域认证协议的前提是终端已经和本地边缘节点 MS1 实现相互认证，并协商出会话密钥 K_{c1} 。跨域认证拟设计思路如下。

Step 1: 终端向外部边缘节点 MS2 发送跨域认证请求，请求中包括自己的匿名身份 SID_u 、随机数 r_2 以及之前与该终端完成本地接入认证的边缘节点 MS1 的身份标识 ID_{ms1} 。

$$C \rightarrow \text{MS2}: \text{Request}(\text{SID}_u \parallel r_2 \parallel \text{ID}_{\text{ms1}}) \quad (18)$$

Step 2: MS2 收到终端的请求后，会秘密保存随机数 r_2' ，然后检查自己信任列表中是否存在 ID_{ms1} ，若没有，MS2 和 MS1 会互相交换数字证书，MS2 在验证 MS1 证书合法性之后从中获取它的公钥 P_{ms1} ，同样，MS1 在验证 MS2 证书的合法性之后会从中获取它的公钥 P_{ms2} ；若 MS2 的信任列表中存在 ID_{ms1} ，则 MS2 直接用 P_{ms1} 加密自己的身份标识 ID_{ms2} 和终端的伪身份标识 SID_u 发给 MS1。

$$\text{MS2} \rightarrow \text{MS1}: P_{\text{ms1}}(\text{SID}_u \parallel \text{ID}_{\text{ms2}}) \quad (19)$$

Step 3: MS1 收到消息后，用私钥解密得到 SID_u 和 ID_{ms2} ，根据 SID_u 从自己的信任列表中找到之前与该终端协商出的会话密钥 K_{c1} ，利用哈希算法计算共享密钥 $K_{\text{cms}} = H(K_{c1}, \text{ID}_{\text{ms2}})$ ，为了保障 K_{cms} 的安全，使用 MS2 的公钥 P_{ms2} 加密 K_{cms} ，这样只有 MS2 可以解开 K_{cms} ；同时，为了确保该消息确实是由 MS1 发出的，将加密后的结果使用 MS1 的私钥进行签名，最后将签名后的结果发给 MS2。

$$\text{MS1} \rightarrow \text{MS2}: \text{Sign}_{\text{ms1}}[P_{\text{ms2}}(K_{\text{cms}} \parallel \text{ID}_{\text{ms2}})] \quad (20)$$

Step 4: MS2 收到 MS1 的消息后，首先使用 MS1 公钥进行验签，若验签通过，则使用自己的私钥解密得到 K_{cms} ；随后 MS2 选取随机数 K_{tmp} 作为与终端之间的临时对称密钥，为了保证 K_{tmp} 安全，使用 K_{cms} 加密 K_{tmp} 、当前时间 T_5 、随机数 r_3 、之前秘密保存的 r_2' 和自己的身份标识 ID_{ms2} 发给终端。

$$\text{MS2} \rightarrow C: K_{\text{cms}}(K_{\text{tmp}} \parallel T_5 \parallel r_2' \parallel r_3 \parallel \text{ID}_{\text{ms2}}) \quad (21)$$

Step 5: 终端收到 MS2 的消息后，首先利用之前与本地边缘节点 MS1 协商出的会话密钥 K_{c1} 和 MS2 的身份标识 ID_{ms2} ，以同样方式计算 K_{cms} ，

然后使用它进行解密, 得到临时对称密钥 K_{tmp} 和 T_5 , 获取当前时间 T_6 , 计算 $T_6 - T_5$ 的结果是否在网络延迟时间内, 若超时, 则认证失败; 否则对比接收到的随机数 r'_2 与之前发给 MS2 的 r_2 是否相同, 当随机数和延迟时间都符合条件时, 完成终端对 MS2 的身份认证。终端选取随机数 K_{c2} 作为新的会话密钥并秘密保存, 获取当前时间 T_7 , 连同接收到的 r'_3 通过 K_{tmp} 加密一起发给 MS2。

$$C \rightarrow \text{MS2}: K_{\text{tmp}}(K_{c2} \parallel T_7 \parallel r'_3) \quad (22)$$

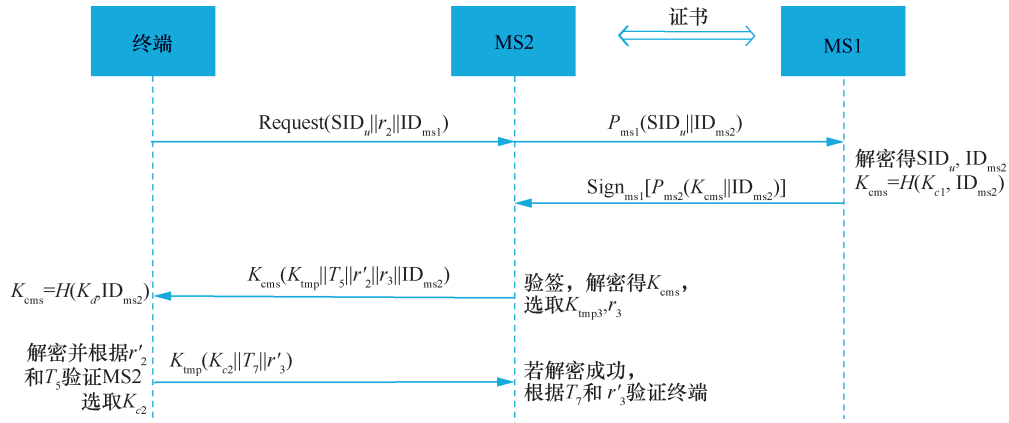


图 5 终端的跨域认证流程

Figure 5 Cross-domain authentication process of terminals

该跨域认证不仅是两个安全域之间的切换, 可推广到多个安全域之间, 当终端从 MS2 所在域移动到新的安全域时, 需要与新的安全域中的边缘节点 MS3 进行跨域认证, 此时, MS2 相当于上述跨域认证过程中的 MS1, MS3 相当于上述跨域认证过程中的 MS2, 以此类推, 实现多个安全域之间的跨域认证。

3 密钥管理方案设计

加密系统的安全性通常不取决于密码算法的机密性, 而是取决于密钥的机密性, 而上述方案中终端的密钥是由 KGC 根据终端身份标识计算产生的, 所以为了保证终端的密钥安全, 要求 KGC 必须是一个可信第三方, 它不能保存终端的私钥, 仅仅保存着终端身份标识 ID_u , 且不能向别人泄露该信息以及计算私钥的方式。当终端身份发生改变, 或者密钥受到威胁需要更换密钥时, 需要向 KGC 申请, 进行身份标识的更新, KGC 需要将新的身份标识添加到标识列表, 并为终端

Step 6: MS2 收到消息后, 首先使用之前选取的 K_{tmp} 进行解密, 若解密失败, 则认证失败; 否则将得到 T_7 、 K_{c2} 和 r'_3 , 计算当前时间 $T_8 - T_7$ 的结果是否在网络延迟时间内, 若超时, 则认证失败; 否则对比之前发给终端的 r_3 与接收到的 r'_3 是否相同, 当网络延迟时间和随机数都符合条件时, 实现 MS2 对终端的身份认证, MS2 秘密保存 K_{c2} , 作为与终端之间的会话密钥。

通过以上步骤实现终端的跨域认证并得到会话密钥 K_{c2} , 整个认证流程如图 5 所示。

计算新的私钥, 同时将原有的身份标识添加到撤销列表。对于已经过期的密钥, 终端仍需秘密保存, 避免被攻击者窃取从而导致从前被加密的信息泄露。

在终端与边缘节点的认证交互过程中, 终端始终以匿名身份 SID_u 进行通信, 所以边缘节点不会知晓终端的真实身份, 只需要保存 SID_u 以及与该终端协商出的会话密钥 K_{c1} , K_{c1} 对于终端的跨域认证影响极大, 若边缘节点 MS 丢失了 K_{c1} 或 K_{c1} 受到威胁需要更新会话密钥时, 需要及时告知终端。首先, MS 向终端发送请求 (request) 以获取会话密钥 K_{c1} , 为了避免被他人冒充, MS 需要对请求进行数字签名, 并附上自己的证书发给终端; 终端首先从证书中获取对方的公钥, 并进行验签, 以确认 MS 身份的真实性, 然后使用该公钥利用 RSA 算法对 K_{c1} 进行加密, 最后发给 MS。MS 使用自己的私钥解密, 得到 K_{c1} 并将它秘密保存在密钥列表中。

4 协议的正确性和安全性分析

身份认证是终端接入网络的第一步，有效的身份认证机制可以在双方通信时验证身份的合法性，保障数据的机密性，为后续通信提供安全保障。本节将对协议的正确性和安全性进行分析。

4.1 终端接入认证协议分析

本文设计的终端与边缘节点之间的接入认证主要是基于 SM9 的数字签名和密钥封装算法，认证过程的关键在于数字签名的生成和验签以及密钥的封装和解封。

4.1.1 协议正确性分析

在数字签名的生成和验签过程中，终端通过 $h = H_2(\text{Msg} \parallel w, N)$ 生成数字签名，边缘节点 MS1 通过计算 $h_2 = H_2(\text{Msg} \parallel w', N)$ ，对比 h_2 与 h 是否一致来完成验签，所以验签的关键是签名生成的 w 和验签时的 w' 是否相同。

在签名时

$$w = g^r = e(P_1, P_{\text{pub-s}})^r = e(P_1, k_s \times P_2)^r = e(P_1, P_2)^{k_s \times r} \quad (23)$$

在验签时

$$\begin{aligned} w' &= ut = \\ e(S', h_1 \times P_2 + P_{\text{pub-s}}) \times g^{h'} &= \\ e(S', h_1 \times P_2 + P_{\text{pub-s}}) \times e(P_1, P_{\text{pub-s}})^{h'} &= \\ e(S', h_1 \times P_2 + k_s \times P_2) \times e(P_1, k_s \times P_2)^{h'} &= \\ e[S', (h_1 + k_s) \times P_2] \times e(P_1, P_2)^{k_s \times h'} &= \\ e([(r - h') \times k_s / (h_1 + k_s) \times P_1, P_2]^{h_1 + k_s}) \times e(P_1, P_2)^{k_s \times h'} &= \\ e(P_1, P_2)^{(r - h') \times k_s} \times e(P_1, P_2)^{k_s \times h'} &= \\ e(P_1, P_2)^{k_s \times r} \end{aligned} \quad (24)$$

从推导结果可以看出， w 和 w' 最终的结果只与系统签名主私钥 k_s 和终端选取的随机数 r 有关，即两端计算出的 w 是相同的，所以终端生成的签名与边缘节点验签的结果也是相等的。

在密钥的封装和解封过程中，边缘节点 MS1 通过 $K_{cl} = \text{KDF}(C \parallel g^r \parallel \text{ID}_{\text{ms1}}, \text{klen})$ 封装会话密钥，终端通过 $K_{cl}' = \text{KDF}(C \parallel w' \parallel \text{ID}_{\text{ms1}}, \text{klen})$ 进行密钥解封，所以若 g^r 与 w' 相等，则解封出的会话密钥一定与封装的相同。在密钥封装时

$$g^r = e(P_{\text{pub-e}}, P_2)^r = e(k_e \times P_1, P_2)^r = e(P_1, P_2)^{r \times k_e} \quad (25)$$

在密钥解封时

$$\begin{aligned} w' &= e(C, e_u) = \\ e(r_1 \times [H_1(\text{SID}_u \parallel \text{hid}_e, N)] P_1 + r_1 \times P_{\text{pub-e}}, e_u) &= \\ e(r_1 \times [H_1(\text{SID}_u \parallel \text{hid}_e, N)] P_1 + r_1 \times k_e \times P_1, e_u) &= \quad (26) \\ e(h_b \times r_1 P_1, k_e / h_b \times P_2) &= \\ e(P_1, k_e / h_b \times P_2)^{h_b \times r_1 \times k_e / h_b} &= e(P_1, P_2)^{r_1 \times k_e} \end{aligned}$$

从上面的推导结果可以看出，只要密文 C 是一样的，解封出的密钥就是一样的，而 C 是经过加密发送的，且解封需要用到的私钥也只有终端知晓，所以终端解封出的密钥是安全的，可以作为二者的会话密钥。

4.1.2 协议安全性分析

(1) 双向认证

本文设计的终端接入认证是利用 SM9 的数字签名算法，终端产生数字签名，边缘节点进行验签，而验签过程需要的 Msg 是经过加密发送的，只有 MS1 可以解开，根据数字签名的不可伪造性，若边缘节点成功验签，则可以验证终端的身份；终端则是通过 MS1 反馈回来的 Msg 和 C 进行验证，若接收到的 Msg 与发送的一致，且后面的密钥可以成功解封，则可以实现对边缘节点的认证，即本文设计的协议可以实现终端与边缘节点的双向认证。

(2) 密钥安全性

终端与边缘节点之间的会话密钥是通过 SM9 的密钥协商算法计算得出的，由上述分析可知，只要密文 C 是一样的，解封出的密钥就是一样的，而密文 C 是经过 K_{sm} 加密发送的， K_{sm} 又是由终端亲自选取并通过 MS1 的公钥加密发送的，只有 MS1 可以解密，其他攻击者无法破解 K_{sm} ，自然无法获取 C ，所以本文协议可以保证密钥的安全性，不会被除边缘节点 MS1 和终端之外的第三方知晓。

(3) 抗假冒攻击

根据上述分析可知，若有攻击者想要冒充终端欺骗 MS1 进行接入认证，由于不知道 Msg 大小，只能自己选取一个待签名消息并利用自己私钥生成签名发给 MS1，MS1 可以验签，但验证的是攻击者的身份，不是终端的，攻击者无法以终端的名义与 MS1 通信；若攻击者想要冒充边缘节点，由于 Msg 是经过 MS1 的公钥加密，只有用

MS1 的私钥可以解密进行验签, 因此, 攻击者无法验签, 更无法进行后续的通信, 所以本文协议可抵抗假冒攻击。

(4) 抗中间人攻击

在本文设计的协议中, 所有的关键信息都是经过加密的, 攻击者即使取到信息也束手无策, 即使破解了消息, 由于攻击者不知道 Msg , 无法验签; 不知道 C 的大小, 无法进行密钥解封; 不知晓 K_{sm} , 无法对后面的消息进行篡改, 因此, 本文协议可以很好地抵抗中间人攻击。

(5) 抗重放攻击

在认证过程中, K_{sm} 是终端选取的一个随机数, Msg 是终端选取的一个待签名信息, 每次通信时, 它们的数值都会不同, 且数字签名的计算过程也加入了随机数, 每次生成的数字签名也不同, 所以攻击者无法对消息进行重放攻击。

(6) 前向安全性

边缘节点在封装会话密钥时, 需要随机选取随机数 r_1 , 所以每次计算出来的会话密钥 K_{c1} 都是不同的, 即使在一次通信过程中密钥被破解, 攻击者也无法解密从前的消息, 具有前向安全性。

综上所述, 本文设计的终端的接入认证具有很好的安全性。

4.2 终端跨域认证协议分析

4.2.1 协议正确性证明

近年来, 协议的正确性证明越来越受到科研界的重视, 而 BAN 逻辑在所有证明方法中最具影响力, 它是一种基于信念的模态逻辑, 可以对协议进行形式化推理, 进而分析协议的正确性。本小节通过 BAN 逻辑推理, 形式化证明跨域认证协议的正确性。BAN 逻辑符号及含义如表 3 所示。

BAN 逻辑推理的原则如下。

- 1) 消息含义规则: $\frac{P \models Q \xrightarrow{k} P, P \triangleleft \{X\}_k}{P \models Q \sim X}$
- 2) 临时校验原则: $\frac{P \models \#(X), P \models Q \sim X}{P \models Q \models X}$
- 3) 消息仲裁原则: $\frac{P \models Q \Rightarrow X, P \models Q \models X}{P \models X}$
- 4) 消息信仰原则: $\frac{P \models Q \models (X, Y)}{P \models Q \models X}$

$$\frac{P \models (X, Y)}{P \models X}$$

$$5) \text{ 消息发送原则: } \frac{P \models Q \sim (X, Y)}{P \models Q \sim X}$$

$$6) \text{ 消息新鲜性原则: } \frac{P \models \#(X)}{P \models \#(X, Y)}$$

表 3 BAN 逻辑符号及含义
Table 3 BAN logical symbols and meanings

符号	含义
$P \models X$	P 相信 X 是真的
$P \triangleleft X$	P 收到消息 X
$P \mapsto X$	P 对消息 X 有管辖权
$P \sim X$	P 发过含有 X 的消息
$\#(X)$	消息 X 是新鲜的
$P \xrightarrow{K} Q \xrightarrow{k} P$	P 和 Q 共享密钥 K 进行通信 k 是 P 的公开密钥
$\{X\}_k$	用密钥 k 加密后的密 X

使用 BAN 逻辑, 将终端与 MS2 之间传输的消息形式化描述如下。

$$\text{M1: request} \parallel \text{ID}_{\text{MS1}} \parallel r_2 \parallel \text{SID}_u$$

$$\text{M2: } \{T_5, r_2', r_3, C \xrightarrow{K_{\text{tmp}}} \text{MS2}, \#(C \xrightarrow{K_{\text{tmp}}} \text{MS2})\}_{K_{\text{cms}}}$$

$$\text{M3: } \{T_7, r_3', C \xrightarrow{K_{c2}} \text{MS2}, \#(C \xrightarrow{K_{c2}} \text{MS2})\}_{K_{\text{tmp}}}$$

所提协议的初始化状态假设如下。

- 1) $C \models \#(T_5)$, $\text{MS2} \models \#(T_7)$
- 2) $\text{MS2} \models C \xleftarrow{K_{\text{cms}}} \text{MS2}$, $\text{MS2} \models C \xleftarrow{K_{c2}} \text{MS2}$
- 3) $C \models C \xleftarrow{K_{\text{cms}}} \text{MS2}$, $C \models C \xleftarrow{K_{c2}} \text{MS2}$
- 4) $C \models \text{MS2} \Rightarrow [T_5, r_2', r_3, C \xleftarrow{K_{\text{cms}}} \text{MS2}, \#(C \xleftarrow{K_{\text{cms}}} \text{MS2})]$
- 5) $\text{MS2} \models C \Rightarrow [T_7, r_3', C \xleftarrow{K_{\text{tmp}}} \text{MS2}, \#(C \xleftarrow{K_{\text{tmp}}} \text{MS2})]$

所提协议要满足的目标如下。

- G1: $C \models \text{MS2} \models [C \xleftarrow{K_{\text{tmp}}} \text{MS2}, \#(C \xleftarrow{K_{\text{tmp}}} \text{MS2})]$
- G2: $C \models \text{MS2} \models [C \xleftarrow{K_{\text{tmp}}} \text{MS2}, \#(C \xleftarrow{K_{\text{tmp}}} \text{MS2})]$
- G3: $\text{MS2} \models C \models [\text{MS2} \xleftarrow{K_{c2}} C, \#(\text{MS2} \xleftarrow{K_{c2}} C)]$
- G4: $\text{MS2} \models [\text{MS2} \xleftarrow{K_{c2}} C, \#(\text{MS2} \xleftarrow{K_{c2}} C)]$

对协议使用 BAN 形式化推理过程如下。

令 $X = [C \xleftarrow{K_{\text{tmp}}} \text{MS2}, \#(C \xleftarrow{K_{\text{tmp}}} \text{MS2})]$ ，
由于 $C \models \#(T_5)$ ，由新鲜性原则可得

$$\frac{C \models \#(T_5)}{C \models \#(X, T_5)} \quad (27)$$

由消息含义规则可得

$$\frac{C \models \text{MS2} \xleftarrow{K_{\text{cms}}} C, C \triangleleft \{T_5, X\}_{K_{\text{cms}}}}{C \models \text{MS2} \sim (T_5, X)} \quad (28)$$

由式(27)、式(28)及临时校验原则可得

$$\frac{C \models \#(T_5, X), C \models \text{MS2} \sim (T_5, X)}{C \models \text{MS2} \models (T_5, X)} \quad (29)$$

由式(29)和仲裁原则可得

$$\frac{C \models \text{MS2} \Rightarrow (T_5, X), C \models \text{MS2} \models (T_5, X)}{C \models (T_5, X)} \quad (30)$$

由式(29)、式(30)和信仰原则可得

$$\frac{C \models \text{MS2} \models (T_5, X)}{C \models \text{MS2} \models X} \quad (31)$$

由式(31)进一步可得

$$\frac{C \models (T_5, X)}{C \models X} \quad (32)$$

同理，令 $Y = [\text{MS2} \xleftarrow{K_{c2}} C, \#(\text{MS2} \xleftarrow{K_{c2}} C)]$ ，
因为 $\text{MS} \models \#(T_7)$ ，由新鲜性原则可得

$$\frac{\text{MS2} \models \#(T_7)}{\text{MS2} \models \#(T_7, Y)} \quad (33)$$

由消息含义规则可得

$$\frac{\text{MS2} \models C \xleftarrow{K_{\text{tmp}}} \text{MS2}, \text{MS2} \triangleleft \{T_7, Y\}_{K_{\text{tmp}}}}{\text{MS2} \models C \sim (T_7, Y)} \quad (34)$$

由式(33)和式(34)以及临时值校验原则可得

$$\frac{\text{MS2} \models \#(T_7, Y), \text{MS2} \models C \sim (T_7, Y)}{\text{MS2} \models C \models (T_7, Y)} \quad (35)$$

由式(35)和仲裁原则可得

$$\frac{\text{MS2} \models C \Rightarrow (T_7, Y), \text{MS2} \models C \models (T_7, Y)}{\text{MS2} \models (T_7, Y)} \quad (36)$$

由式(35)、式(36)和信仰原则可得

$$\frac{\text{MS2} \models C \models (T_7, Y)}{\text{MS2} \models C \models Y} \quad (37)$$

由式(37)进一步可得

$$\frac{\text{MS2} \models (T_7, Y)}{\text{MS2} \models Y} \quad (38)$$

综上所述，完成协议的安全性证明。

4.2.2 协议安全性分析

(1) 双向身份认证

本文设计的终端跨域认证的前提是终端首先与本地边缘节点 MS1 实现接入认证，并协商出会话密钥 K_{c1} ，然后利用 Hash 函数计算新的会话密钥 K_{c2} ，在跨域认证过程中加入随机数 r_2 、 r_3 和时间戳，只有当随机数对比一致且网络延迟时间在合理范围内才算认证成功，最终实现了彼此的双向认证。

(2) 密钥安全性

认证过程中， K_{cms} 作为共享密钥，只有 MS1 和终端可以计算得到，MS1 会将 K_{cms} 加密后通过安全信道发给 MS2，只有 MS2 的私钥可以解密，所以 MS2 接收到的 K_{cms} 是安全的；由 MS2 选取临时对称密钥 K_{tmp} 并使用 K_{cms} 加密后发给终端，由于其他人无法解密 K_{cms} ，所以终端收到的 K_{tmp} 可以确保数据的完整性，最终由终端秘密选取会话密钥 K_{c2} ，并使用 K_{tmp} 加密发给 MS2，整个过程环环相扣，窃听者无法解密其中的任何一环，从而保障会话密钥 K_{c2} 的安全性。

(3) 抗假冒攻击

K_{cms} 是由 MS1 计算并使用 MS2 的公钥加密发给 MS2 的，这意味着只有 MS2 可以解密，若有人想冒充 MS2 欺骗终端，由于缺少 K_{cms} ，它是无法使用 K_{cms} 与终端进行通信的；同理，若有人想冒充终端向 MS2 发送认证请求，由于缺少 K_{c1} ，即使收到了 MS2 的反馈也因无法计算 K_{cms} 从而无法解密，可见，任何攻击者都无法在中间冒充终端或者 MS2 进行认证。

(4) 中间人攻击

在认证过程中，除了第一步外，其他交互信息都是经过加密的，攻击者不知道 K_{cms} 或者 K_{tmp} ，所以即使截取到消息也无法解密；若它想窃取会话密钥 K_{c2} ，必须先破解 K_{tmp} ，要想破解 K_{tmp} ，必须要知晓 K_{cms} ，然而 K_{cms} 是用 P_{ms2} 公钥加密发送给 MS2 的，只有 MS2 可以解密，因此，窃听者无法从密文破解出共享密钥 K_{cms} ，也就无法知晓

K_{c2} , 更无法对后续消息进行篡改; 若攻击者篡改了第一步的随机数, 则后续的认证将失败, 所以本文设计的终端跨域认证可以很好地抵抗中间人攻击。

(5) 抗重放攻击

在认证过程中, 加入了时间戳和随机数, 可以保证终端与 MS2 每次通信时的消息都是新鲜的, 并且双方都会记住之前使用过的随机数, 攻击者是无法通过之前的消息进行重放攻击的。

(6) 前向安全性

在认证过程中, 每一次的临时对称密钥 K_{tmp} 和会话密钥 K_{c2} 都是随机选取的, 并且是双方秘密选取并通过安全信道加密传输, 这样可以降低密钥被预测的可能性, 即使一次通信过程中会话密钥被窃取, 攻击者也无法破解从前的消息。

(7) 用户匿名

在本文方案中, 用户的身份 ID_u 是不会公开传输的, 而是在注册阶段利用 Hash 函数计算一个匿名身份标识 SID_u , 此后传输使用的都是匿名身份标识, 根据 Hash 函数的单向安全性, 任何人都无法破解用户的身份, 从而保障用户身份的匿名性。

为了清楚地展示本文设计的跨域认证协议在安全方面的性能, 将本文方案与文献[5]方案、文献[22]方案和文献[23]方案进行对比, 结果如表4所示。

在文献[5]方案中, 终端与边缘节点 MS2 的会话密钥由 MS2 和 MS1 共同计算得出, 并由 MS2 发给终端, 后续在终端与 MS2 通信过程中, MS1 由于知晓该会话密钥有可能成为中间攻击者窃取消息, 所以该方案的密钥不是十分安全的, 无法抵抗中间人攻击; 在文献[22]方案中, 所有的随机数都是在公共信道上传输的, 攻击者可以截取该随机数进行修改, 然后将消息进行重放攻击, 所以该方案无法抵抗重放攻击, 也不具备单点登录的功能, 而且在认证过程中加入双线性配对, 计算成本显著增加; 在文献[23]方案中, 终端发送给代理服务器的消息没有经过加密, 存在有中间攻击者截取消息然后冒充代理服务器继续通信的可能, 无法抵抗假冒攻击, 而且没有给出逻辑上的安全性证明, 同时不具备实现单点登录的功能; 而本文设计的终端跨域认证协议可以有效抵抗以上攻击, 具备较好的安全性。

表4 跨域认证方案安全性对比
Table 4 Security comparison of cross domain authentication Schemes

安全性能	文献[5] 方案	文献[22] 方案	文献[23] 方案	本文 方案
双向认证	√	√	√	√
密钥安全	×	√	√	√
中间人攻击	×	√	√	√
抗重放攻击	√	×	√	√
抗假冒攻击	√	√	×	√
前向安全	√	√	√	√
单点登录	×	×	×	√
用户匿名	√	√	√	√
安全可证	√	√	×	√

5 性能分析

现有的关于边缘计算环境下认证的研究大部分是接入认证, 关于跨域认证的研究很少, 而本文的创新点主要是设计一种边缘计算环境下轻量安全的终端跨域认证协议, 所以在协议的性能分析部分, 主要考虑终端与边缘节点之间的跨域认证。本小节将从协议的计算成本和通信成本两方面对所提出的跨域认证协议进行性能分析。

5.1 计算成本

计算成本是指终端和边缘节点在认证过程中执行加解密、哈希运算等操作耗费的时间, 为了分析方案的计算成本, 需要统一设置各操作的参数, 如表5所示。

表5 各操作的参数
Table 5 Parameter of each operation

操作	参数
非对称加密	T_{re}
非对称解密	T_{rd}
对称加密	T_{ae}
对称解密	T_{ad}
哈希操作	T_h
点加操作	T_{add}
点乘操作	T_{mul}
双线性对操作	T_{pair}

在实验中, 通过本地的计算机来模拟终端, 平台参数为 Gen Intel Core i5-1135G7 @ 2.40 GHz, 内存为 4 GB, 采用 Ubuntu 20.04 系统; 使用阿里

云服务器来模拟边缘节点, 平台参数为 Intel Xeon Platinum 8269CY CPU @ 2.50 GHz, 4 GB 内存, 采用 Ubuntu 20.04.3 系统。

在计算各操作执行时间时, 非对称加密使用的是 RSA 算法, 使用最小的 1 024 bit 密钥, 待加密的明文长度为 117 byte, 实验中通过公钥加密、私钥解密实现非对称加密操作, 数字签名使用的是 RSA 算法; 对称加密使用的是 AES 算法, 使用最小的 128 bit 密钥, 通过 CBC 模式对 256 bit 的数据进行分组加密, 由于 CBC 模式必须按顺序对数据块加密而解密时可以对数据块并行解密, 所以实验中对称加密的时间比解密时

间长一些。

哈希运算使用的是 SHA-256 算法, 它可以任意长度的数据转化为 256 bit 的字符串; 双线性配对、点乘和点加运算需要椭圆曲线的参与, 实验中选取的椭圆曲线为 Type-A 类型, 它是一种质数阶双线性群, 曲线的方程为 $y^2 = x^3 + x$, 关于曲线的参数可参考官方文档; 整个实验使用 Java 语言结合 JPBC 密码学库完成, 为了更加客观准确地分析计算成本, 实验中使用纳秒级别的时间计数器进行测试, 同时对每种基本操作循环测试 1 000 次, 取平均值作为最终的结果。在两种平台上执行以上基本操作的执行时间如表 6 所示。

表 6 基本操作的执行时间
Table 6 Execution time of basic operations

平台	T_{re}	T_{rd}	T_{ac}	T_{ad}	T_h	T_{pair}	T_{mul}	T_{add}
终端	0.741	1.596	0.118	0.063	0.018	7.461	2.304	0.106
MS2	0.554	1.011	0.085	0.041	0.013	4.658	1.925	0.036

为了更好地评估协议计算成本, 将本文设计的终端认证方案与文献[5]方案、文献[22]方案以及文献[23]方案进行对比, 其中, 文献[5]方案设计了一种在边缘计算环境下终端与外部边缘节点 MS2 间的跨域认证方案; 文献[22]方案设计了一种在云计算环境下终端与外部代理服务器 FA 之间的跨域认证方案; 文献[23]方案设计了一种在雾计算环境下终端与雾节点之间的认证方案。这 3 种方案的应用场景比较相似, 认证过程中使用的是密码学中常见的加解密技术, 均需要依赖第三方进行实现, 但实现的方式各有不同, 并且以上 3 种方案在各自的应用领域都是较为优秀的, 具备一定的前沿性和代表性, 所以将本文方案与以上 3 种方案进行对比。

在文献[5]方案中, 终端需要使用 1 次对称加密 T_{ac} 向边缘节点 MS2 发送认证请求, 2 次对称解密 T_{rd} 从 MS2 解出会话密钥; 边缘节点 MS2 需要利用椭圆曲线执行 2 次点乘 T_{mul} 计算会话密钥、一次非对称加密 T_{re} 把部分参数发给可信第三方, 1 次非对称解密 T_{rd} 从可信第三方解密消息以及 1 次对称加密 T_{ac} 把会话密钥发给终端。即终端所需操作时间为 $T_{ac} + 2T_{ad}$, 边缘节点 MS2 所需操作时间为 $T_{ac} + T_{re} + T_{rd} + 2T_{mul}$ 。

在文献[22]方案中, 移动终端首先执行 3 次哈希 T_h 检查参数是否相等, 然后执行 2 次点乘 T_{mul} , 2 次 T_h 和 1 次对称加、解密操作计算相关参数, 将结果发给外部代理服务器 FA; FA 执行 2 次点乘 T_{mul} 和 2 次 T_h 将结果发给第三方 HA, FA 从 HA 接收到消息后还需要执行 3 次 T_h 和 1 次点乘 T_{mul} 操作计算会话密钥并将结果发给终端; 终端最后还需要执行 4 次 T_h 和 2 次点乘 T_{mul} 向外部代理服务器 FA 发送反馈。即终端所需操作时间为 $9T_h + 4T_{mul} + T_{ac} + T_{ad}$, FA 所需操作时间为 $3T_{mul} + 5T_h$ 。

在文献[23]方案中, 终端需要首先执行 2 个点乘 T_{mul} 和 2 个哈希 T_h 操作向代理服务器 FN 发送消息, FN 需要执行 3 个点乘 T_{mul} 和 2 个哈希 T_h 操作, 将消息发给第三方, 并执行 1 个点乘 T_{mul} 和 2 个哈希 T_h 处理从第三方接收消息, 发给终端; 终端最后执行一个点乘和 2 个 T_h 操作完成认证。即终端所需操作时间为 $3T_{mul} + 4T_h$, FN 所需操作时间为 $4T_{mul} + 4T_h$ 。

而本文设计的跨域认证方案只需要终端执行 1 次 T_h 计算 K_{cms} 、1 次对称解密 T_{ad} 得到 K_{tmp} 以及 1 次对称加密 T_{ac} 向 MS2 发送反馈; 在边缘节点处, 只需要 MS2 执行 1 次非对称加、解密和 1 次验签用来与 MS1 交互, 以及 1 次对称加、解密用来与

终端交互。即终端所需操作时间为 $T_h + T_{ac} + T_{ad}$ ，MS2 所需操作时间 $T_{ac} + T_{ad} + T_{re} + 2T_{rd}$ 。

假设以上方案均使用本文实验中的加解密算法和曲线参数，各基本操作执行的时间也以表 5 中数据为标准，基于上述分析，整理出 4 种方案在终端认证过程中的计算成本对比如表 7 所示，由于其他几种方案在第三方的执行过程中均使用了多次点乘运算，远远大于本文第三方 MS1 的计算成本，所以在比较计算成本时省去了第三方的比较。

表 7 计算成本对比
Table 7 Comparison of calculation cost

方案	终端操作时间/ms	MS2/FA/FN 操作时间/ms	总时间/ms
文献[5] 方案	$T_{ac} + 2T_{ad} = 0.244$	$T_{ac} + T_{re} + T_{rd} + 2T_{mul} = 5.500$	5.744
文献[22] 方案	$9T_h + 4T_{mul} + T_{ac} + T_{ad} = 9.559$	$5T_h + 3T_{mul} = 5.840$	15.399
文献[23] 方案	$3T_{mul} + 4T_h = 6.984$	$4T_{mul} + 4T_h = 7.752$	14.736
本文 方案	$T_h + T_{ac} + T_{ad} = 0.199$	$T_{ac} + T_{ad} + T_{re} + 2T_{rd} = 2.702$	2.901

从表中可以看出，4 种方案都没有使用代价高昂的双线性配对操作，然而点乘操作耗费的时间也很多，在终端，文献[22]方案和文献[23]方案由于使用了多次 T_{mul} ，所以耗费的时间比其他两种方案多很多，而文献[5]方案与本文方案只需要普通的加解密操作，所以耗费时间较少，并且本文方案比文献[5]方案在终端少一次对称加密 T_{ad} ，使得终端的计算成本更低；在边缘服务器端，其他 3 种方案均使用了点乘操作，计算时间均比本文方案多很多。基于上述分析，不管是在终端还是在边缘服务器端，本文提出的身份认证协议都具有更低的计算成本，大大减轻了终端的计算压力。

5.2 通信成本

通信成本主要是指终端和边缘节点在认证过程中进行网络传输的成本，简单来说就是发送消息的长度，消息长度越短，网络传输的通信成本越低。为了比较不同方案的通信成本，本文统一设置通信过程中各参数的表示及消息长度，如表 8 所示。

表 8 参数含义及消息长度取值
Table 8 Parameter meanings and length value

参数	含义	消息长度/bit
P	椭圆曲线大素数	512
ID	身份标识	32
T	时间戳	256
Z_q / r	域及域中随机数	160
G	椭圆曲线上的点	1 024
M_{ac}	消息验证码	160
H	哈希函数输出的消息	256
K	会话密钥	256
M_{rsa}	非对称加密后的消息	1 024
M_{aes}	对称加密后的消息	256

在计算通信成本的实验中，选取的加解密算法以及曲线参数与上述计算成本的算法及参数相同，椭圆曲线依旧选择的是 Type-A 类型，方程为 $y^2 = x^3 + x$ ，在官方手册中给出了它的两个参数，其中，曲线阶数 $p=512$ bit，域 $Z_q=160$ bit，所以本文设置椭圆曲线的 $p=512$ bit，这意味着曲线上点 G 的长度是 1 024 bit；随机数 r 是从 Z_q 域中选取，所以设置 r 的长度为 160 bit；使用 SHA-256 算法，所以哈希操作输出的消息长度 $H=256$ bit；非对称加密使用 1 024 bit 密钥的 RSA 算法，公钥加密后的结果 M_{rsa} 取 1 024 bit；对称加密使用 256 bit 密钥的 AES 算法，加密后的结果 M_{aes} 长度取 256 bit；终端及边缘节点的身份标识均为 256 bit，实验中的时间戳 T 长度取值为 32 bit，生成的会话密钥 K 取作 256 bit。综上所述，将以上取值进行整理，如表 8 所示。

为了更好地评估协议的通信成本，仍将本文方案与文献[5]方案、文献[22]方案以及文献[23]方案进行对比。

在文献[5]方案中，终端需要先发送 1 个对称加密后的消息 M_{aes} 、3 个 ID 和 1 个随机数 r 发给 MS2，MS2 将非对称加密后的结果 M_{rsa} 发给 MS1，MS1 执行相应计算，然后向 MS2 发送 1 个非对称加密后的结果 M_{rsa} ，最后 MS2 向终端发送 2 个对称加密的结果 M_{aes} 。即终端需要发送的消息为 $|M_{aes}| + 3|ID| + |r|$ ，边缘节点需要发送的消息为 $2|M_{aes}| + |M_{rsa}|$ 。

在文献[22]方案中，终端需要先向外部代理

服务器 FA 传输 1 个对称加密的结果 M_{aes} 、1 个哈希结果 H 、1 个 G 、1 个消息验证码 M_{ac} 和 1 个 T ；FA 把收到的消息原封不动地发给第三方 HA 后，再发送一个 G 、一个 M_{ac} 和一个时间 T ，最后 FA 向终端传输 1 个 G 、2 个 M_{ac} 和 2 个时间 T ；终端收到消息后再向 FA 传输 1 个消息验证码 M_{ac} 和 1 个时间戳 T 完成认证。即终端需要发送的消息为 $|M_{aes}| + 2|M_{ac}| + |G| + 2|T| + |H|$ ，FA 需要发送的消息为 $|M_{aes}| + 4|M_{ac}| + 3|G| + 4|T| + |H|$ 。

在文献[23]方案中，终端首先向代理服务器 FN 传输 1 个 G 、1 个哈希结果 H 、1 个随机数和 1 个 T ，代理服务器 FN 需要向第三方传输 2 个哈希结果 H 、3 个 G 、2 个时间 T 和 2 个随机数 r ，并从第三方接收 3 个 G 、2 个哈希结果 H 和 1 个 T ，最后向终端传输 3 个 G 、1 个时间 T 和 1 个 H 完成认证。即终端需要发送的消息为 $|G| + |H| + |T| + |Z_q|$ ，FN 需要发送的消息为 $6|G| + 3|H| + 3|T| + 2|Z_q|$ 。

在本文设计的跨域认证方案中，终端首先需要传输 2 个 ID 和一个随机数 r 给 MS2，MS2 向 MS1 传输一个非对称加密的结果 M_{rsa} ，MS1 执行加解密和签名等操作，由于 RSA 数字签名是需要发送原文的，所以 MS1 需要传输 2 个非对称加密的消息 M_{rsa} 给 MS2，MS2 解密验签后需要传输 1 个对称加密后的结果 M_{aes} 给终端，终端最后需要向 MS2 传输 1 个 M_{aes} 完成认证。即终端需要发送的消息为 $2|ID| + |r| + |M_{aes}|$ ，边缘节点需要发送的消息为 $|M_{aes}| + |M_{rsa}|$ 。

假设以上方案在通信过程中均使用本文中的加、解密算法和曲线参数，各参数和消息长度也以表 8 中的数据为标准，基于上述分析，整理出 4 种方案在终端跨域认证过程中的通信成本对比如表 9 所示。由于文献[22]方案和文献[23]方案在第三方的通信成本远大于本文方案在第三方 MS1 处的通信成本，所以在比较通信成本时对第三方的通信成本不再考虑。

表 9 通信成本对比
Table 9 Comparison of communication cost

方案	终端发送消息长度/bit	MS2/FA/FN 发送消息长度/bit	总计/bit
文献[5]方案	$ M_{aes} + 3 ID + r = 1\ 184$	$2 M_{aes} + M_{rsa} = 1\ 536$	2 720
文献[22]方案	$ M_{aes} + 2 M_{ac} + G + 2 T + H = 1\ 920$	$ M_{aes} + 4 M_{ac} + 3 G + 4 T + H = 4\ 192$	6 112
文献[23]方案	$ G + H + T + Z_q = 1\ 472$	$6 G + 3 H + 3 T + 2 Z_q = 7\ 328$	8 800
本文方案	$2 ID + r + M_{aes} = 928$	$ M_{aes} + M_{rsa} = 1\ 280$	2 208

从表中可以看出，文献[22]方案和文献[23]方案由于使用了点乘运算使得生成的消息长度显著增加，通信成本远高于文献[5]方案和本文方案；和文献[5]方案相比，本文方案在终端通信时可以少传输一个 ID，在 MS2 通信时可以少传输一个对称加密的结果 M_{aes} ，使得本文方案的通信成本大大降低，减轻了网络传输压力。

最后，为了更加清楚直观地展示本文方案的性能，将计算成本和通信成本结合在一起进行综合分析，如图 6 所示，柱状图代表计算成本，折线图代表通信成本。可以看出，文献[23]方案虽然比文献[22]方案节省了一点计算成本，但通信成本却比文献[22]方案高很多，这种牺牲通信成本换取计算成本的方式并不可取；而本文方案无

论是计算成本还是通信成本，均比其他方案降低了很多，性能上优于其他方案。

综上所述，本文设计的终端跨域认证协议在终端处不需要进行双线性配对和点乘等复杂性操作，只需要执行简单的哈希运算和加、解密操作，降低了终端的计算成本；通信过程中无须传输复杂的信息，减少了通信成本；边缘节点不需要存储终端的其他信息，而终端也只是需要存储 K_{c1} 便可以进行跨域认证，具有较低的存储成本；且认证过程中终端使用的都是匿名传输，不会暴露真实身份，保护了终端用户的身份隐私，在满足较高安全性的前提下，本文设计的终端跨域认证协议更能满足资源受限的终端设备需求，是一种轻量安全的认证协议。

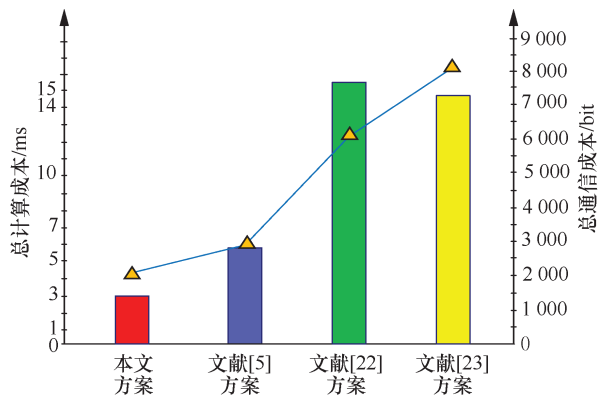


图6 方案总体性能对比

Figure 6 Overall performance comparison of the scheme

6 结束语

针对边缘计算环境下终端设备数量多、移动性强、资源受限的问题,本文设计了一种轻量级的终端跨域认证协议。使用国密 SM9 算法实现了终端与本地边缘节点 MS1 的接入认证,并协商出会话密钥 K_{c1} ,之后当终端移动到新的安全域时,借助 K_{c1} 与新的安全域中的边缘节点 MS2 进行跨域认证,并协商出新的会话密钥 K_{c2} 。通过 BAN 逻辑推理形式化证明了所提方案的正确性,并对协议的安全性进行了详细分析,最后在计算成本和通信成本两方面与现有的认证方案进行对比,结果表明,本文设计的终端跨域认证协议显著降低了终端设备的计算压力,减少了通信成本,更满足资源受限的终端设备需求。

参考文献:

- [1] 沈传年. 边缘计算安全与隐私保护研究进展[J]. 网络安全与数据治理, 2022, 41(8): 41-48.
SHEN C N. Research progress of edge computing security and privacy protection [J]. Network Security and Data Governance, 2022, 41 (8): 41-48.
- [2] LIU L, WANG Q, LI B Q. A system architecture for intelligent agriculture based on edge computing[J]. International Journal of Computer Applications in Technology, 2020, 64(2): 126-132.
- [3] YANG C, HUANG Q, LI Z, et al. Big data and cloud computing: innovation opportunities and challenges[J]. International Journal of Digital Earth, 2017, 10(1): 13-53.
- [4] WANG T, QIU L, SANGAIAH A K, et al. Edge-computing-based trustworthy data collection model in the internet of things[J]. IEEE Internet of Things Journal, 2020, 7(5): 4218-4227.
- [5] 吴卫. 边缘计算环境下物联网身份认证与隐私保护技术研究[D]. 西安: 西安电子科技大学, 2019.
WU W. Research on IoT identity authentication and privacy protection technology in edge computing environment[D]. Xi'an: Xidian University, 2019.
- [6] YU Y. Mobile edge computing towards 5G: vision, recent progress, and open challenges[J]. China Communications, 2016, 13: 89-99.
- [7] YANG L, LIU J. Cross domain authentication based on blockchain for mobile terminals in edge computing environment[C]//2021 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE). 2021: 525-529.
- [8] 马立川, 裴庆祺, 肖慧子. 万物互联背景下的边缘计算安全需求与挑战[J]. 中兴通讯技术, 2019, 25(3): 37-42.
MA L C, PEI Q Q, XIAO H Z. Security requirements and challenges of edge computing in the context of the internet of everything[J]. ZTE Communication Technology, 2019, 25(3): 37-42.
- [9] ISLAM A, DEBNATH A, GHOSE M, et al. A survey on task offloading in multi-access edge computing[J]. Journal of Systems Architecture, 2021, 118: 102225.
- [10] AHMED E, REHMANI M H. Mobile edge computing: opportunities, solutions, and challenges[J]. Future Generation Computer Systems, 2017, 70: 59-63.
- [11] WANG X, GAO F, ZHANG J, et al. Cross-domain authentication mechanism for power terminals based on blockchain and credibility evaluation[C]//2020 5th International Conference on Computer and Communication Systems (ICCCS). 2020: 936-940.
- [12] FAN K, PAN Q, WANG J, et al. Cross-domain based data sharing scheme in cooperative edge computing[C]//2018 IEEE International Conference on Edge Computing (EDGE). 2018: 87-92.
- [13] CHEN C L, CHIANG M L, HSIEH H C, et al. A lightweight mutual authentication with wearable device in location-based mobile edge computing[J]. Wireless Personal Communications, 2020, 113: 575-598.
- [14] SUN H, TAN Y, LI C, et al. An edge-cloud collaborative cross-domain identity-based authentication protocol with privacy protection[J]. Chinese Journal of Electronics, 2022, 31(4): 721-731.
- [15] LYU P, WANG Y, WANG Y Z, et al. A highly reliable cross-domain identity authentication protocol based on blockchain in edge computing environment[C]//2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design(CSCWD). 2022: 1040-1046.
- [16] 张金花, 李晓伟, 曾新, 等. 边缘计算环境下基于区块链的跨域认证与密钥协商协议[J]. 信息安全学报, 2021, 6(1): 54-61.
ZHANG J H, LI X W, ZENG X, et al. Cross domain authentication and key agreement protocol based on blockchain in edge computing environment[J]. Journal of Information Security, 2021, 6 (1): 54-61.
- [17] DONG S, YANG H, YUAN J, et al. Blockchain-based cross-domain authentication strategy for trusted access to mobile devices in the IoT[C]//2020 International Wireless Communications and Mobile Computing (IWCMC). 2020: 1610-1612.
- [18] WANG W, HU N, LIU X. BlockCAM: a blockchain-based cross-domain authentication model[C]//2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). 2018: 896-901.
- [19] 赖建昌, 黄欣沂, 何德彪, 等. 国密 SM9 数字签名和密钥封装算法的安全性分析[J]. 中国科学: 信息科学, 2021, 51(11): 1900-1913.
LAI J C, HUANG X Y, HE D B, et al Security analysis of digital signature and key encapsulation algorithm for state secret SM9[J]. Chinese Science: Information Science, 2021, 51 (11): 1900-1913.

- [20] 殷明. 基于标识的密码算法 SM9 研究综述[J]. 信息技术与信息化, 2020(5): 88-93.

YIN M. Overview of research on identity based cryptography algorithm SM9[J]. Information Technology and Informatization, 2020 (5): 88-93

- [21] 丁洁, 林志阳, 沈获帆. 浅谈无可信第三方的身份认证协议[J]. 科技创新与应用, 2016(10): 57-58.

DING J, LIN Z Y, SHEN D F. On the identity authentication protocol without trusted third parties[J]. Technological Innovation and Application, 2016 (10): 57-58

- [22] LU Y, XU G, LI L, et al. Robust privacy-preserving mutual authenticated key agreement scheme in roaming service for global mobility networks[J]. IEEE Systems Journal, 2019, 13(2): 1454-1465.

- [23] MA M, HE D, WANG H, et al. An efficient and provably secure authenticated key agreement protocol for fog-based vehicular ad-hoc networks[J]. IEEE Internet of Things Journal, 2019, 6(5): 8065-8075.



张新有(1971-), 男, 河南三门峡人, 博士, 西南交通大学副教授, 主要研究方向为分布式计算与应用、网络安全。



邢焕来(1984-), 男, 河北唐山人, 博士, 西南交通大学副教授, 主要研究方向为人工智能、网络安全。

[作者简介]



朱宏颖(1997-), 女, 辽宁抚顺人, 西南交通大学硕士生, 主要研究方向为边缘计算、网络安全。



冯力(1974-), 男, 四川广元人, 博士, 西南交通大学教授、博士生导师, 主要研究方向为人工智能、网络安全。