IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

出版日期 xxxx 00, 0000，当前版本日期 xxxx 00, 0000。

数字对象标识符10.1109/ACCESS.2017.DOI

Fabric-iot：物联网中基于区块链的访问控制系统

韩柳[1], 韩德志[1], 李敦.[1]
[1]上海海事大学信息工程学院，上海，中国

通讯作者。Dezhi Han (e-mail: dzhan@shmtu.edu.cn)

这项工作得到了国家自然科学基金61672338和61873160的部分支持。

ABSTRACT

物联网设备具有一些特殊的特性，如移动性、有限的性能和分布式部署，这使得传统的集中式访问控制方法难以支持当前大规模物联网环境中的访问控制。为了应对这些挑战，本文提出了一个名为fabric-iot的物联网访问控制系统，该系统基于Hyperledger Fabric区块链框架和基于属性的访问控制（ABAC）。该系统包含三种智能合约，即设备合约（DC）、政策合约（PC）和访问合约（AC）。DC提供了一种存储设备产生的资源数据的URL的方法，以及一种查询方法。PC提供了为管理用户管理ABAC策略的功能。AC是为普通用户实现访问控制方法的核心程序。结合ABAC和区块链技术，fabric-iot可以在物联网中提供去中心化、细粒度和动态的访问控制管理。为了验证该系统的性能，设计了两组仿真实验。结果表明，fabric-iot可以在大规模的请求环境中保持高吞吐量，并在分布式系统中有效地达成共识，确保数据的一致性。

索引词 区块链、物联网、ABAC、Hyperledger Fabric、分布式系统。

I. 简介

随着互联网和计算机硬件的发展，越来越多的设备被连接在一起。通过无线网络，使物联网（IoT）的规模越来越大。物联网是一个由大量的传感器和网关组成的分布式网络。物联网设备一直在与环境互动，产生不同类型的数据资源，如图像、音频、视频、数字信号等。物联网的所有系统和应用都可以接入互联网，有效地共享资源和信息[1]。也就是说，我们正生活在一个万物互联的世界里[2]。然而，由于物联网设备的分布式部署及其数量和规模庞大，设备资源的访问控制正面临着巨大的挑战。物联网设备产生的资源往往包含隐私和敏感数据，因此一旦被非法获取，将产生严重后果。访问控制技术是保护资源的重要手段，已被广泛应用于各种系统和环境[3]。传统的访问控制方法包括全权访问控制（DAC）、基于身份的访问控制（IBAC）和强制访问控制（MAC）等。但这些方法都是集中式设计，有

的缺点是单点故障，难以扩展，可靠性低，吞吐量低。事实上，物联网设备可能属于不同的组织或用户，而且很可能具有移动性和有限的性能，这使得集中式访问控制难以满足物联网环境下的访问控制要求。基于属性的访问控制（ABAC）是一种逻辑访问控制模型，它根据条目、操作和相关环境的属性来控制主体和对象之间的访问[4]。ABAC首先分别提取用户（主体）、资源（对象）、权限和环境的属性，然后灵活地组合这些属性的关系，最后将权限的管理转化为属性的管理，提供一种细粒度、动态的访问管理方法。区块链[5]是另一种新兴的数据管理技术，它通过分布式存储保证数据的可靠性。数据的读取或修改记录作为交易写入区块，区块之间通过哈希算法连接成链，以保证数据的完整性。它通过P2P网络和共识算法实现节点间的数据同步，使参与区块链网络的各方达成共识，从而保证数据的一致性。

Hyperledger Fabric[6]是一个开源的区块链开发平台，它不仅具有区块链的特点，如去中心化账本、不可变、群体共识等，还提供更高效的共识机制、更高的吞吐量、智能合约，以及对多个组织和账本的支持。

在本文中，我们将区块链技术应用于物联网访问控制，设计并实现了一个名为fabic-iot的访问控制系统，该系统是基于Hyperledger Fabric和ABAC。通过使用分布式架构，fabic-iot可以追踪记录，提供动态访问控制管理，解决物联网的访问控制问题。

本文的主要贡献在于。

1) 我们根据现实生活中物联网设备的数据生产情况，定义了一个设备资源共享模型。该模型使设备产生的数据资源与URL一一对应，大大简化了设备资源的共享模式和存储结构。

2) 我们提出了一个基于区块链的物联网访问控制系统，名为fabric-iot，并详细描述了其工作流程和架构。该系统使用分布式架构来分离用户和设备，并实现了权限的动态管理，以支持高效访问。

3) 我们设计了三种基于Hyperledger Fabric平台的智能合约。第一种实现了ABAC模型。第二个实现了ABAC策略管理。最后一个实现了设备资源管理。

4) 我们详细介绍了fabric-iot的网络初始化、Chaincode in- stallation和智能合约调用。

5) 我们设计了两组对比实验来验证系统性能和共识速度。

本文的其余部分组织如下。第二节介绍了相关工作。第三节详细介绍了Hyperledger Fabric的结构和运行机制，以及ABAC模型。第四节介绍了资源模型、策略模型、系统结构、工作流程和智能合约的实现。在第五节中，我们展示了如何建立fabric-iot系统，以及如何使用它来管理物联网资源的访问控制。我们设置了两组对比实验，然后分析结果。在第六节中，我们对本文进行了总结，并对未来的工作进行了展望。

II. 相关的工作

物联网的快速发展要求对分布式访问控制有更高的标准。区块链技术在评估控制方面有四个优势，即去中心化、数据加密、可扩展性和不可篡改。目前，区块链技术已经发展到3.0版本。作为其核心技术，智能合约为应用程序构建了一个安全可靠的运行环境，并赋予区块链更强大的功能。因此，在现有访问控制方法的基础上，许多学者通过与区块链和智能合约的结合，提出了多种物联网访问控制方法。

参考文献[7]提出了一种基于Fabric的安全数据传输技术，实现了工业物联网的电力交易中心，解决了安全性低、管理成本高、监管难度大的问题。参考文献[8]中，提出了一种物联网设备的去中心化访问控制方案。该方案使用单一的智能合约来减少节点之间的通信成本。一个被称为管理中心的节点被设计用来与设备互动，避免了区块链和物联网设备之间的直接互动。它有六个优点，分别是移动性、可访问性、并发性、轻量级、可扩展性和跨透明性。在参考文献[9]中，提出了一个基于Ethereum智能合约的访问控制方案，其中包括三个智能合约：访问控制合约（ACC）、判断合约（JC）和注册合约（RC）。ACC通过检查对象的行为来实现基于策略的授权。JC用于判断错误行为并返回相应的惩罚。RC用于注册上述两个智能合约并提供更新、删除和其他操作。最后，该架构由一台PC、一台笔记本和两台Raspberry Pi实现。在参考文献[10]中，提出了一个基于Ethereum的分布式应用（DAPP）。它将SaaS商业模式与区块链相结合，用于买卖传感器数据。参考文献[11]提出了一种改进的区块链结构。在企业网络中，私有链被用来以分布式方式管理物联网的设备配置文件。它将设备配置文件存储在区块链上，并通过智能合约监控操作。在参考文献[12]中，路由器节点的路由信息被保存到区块链上，以确保路由信息不能被篡改和追踪。在参考文献[13]中，提出了一种基于属性的物联网访问控制方法，它通过区块链保存属性数据。这种方法避免了数据被篡改，简化了访问控制协议，以满足物联网设备的计算能力和能源限制。在参考文献[14]中，提出了一个多区块链分布式密钥管理架构（BDKMA），并引入了雾计算方法来减少多链操作的延迟，可以更好地保证用户的隐私和安全。在参考文献[15]中，区块链被用于车联网的信任管理。通过结合工作证明（PoW）和股权证明（PoS）两种共识机制，可以动态改变挖矿节点的难度，从而更有效地达成共识。参考文献[16]提出了一种基于雾计算的高效、安全的路况监测认证方案。参考文献[17]提出了一种边缘链，利用区块链的货币体系，将边缘计算资源与账户和物联网设备之间的资源使用情况联系起来。边缘链建立了一个基于设备行为的信任模型，以控制物联网设备从边缘服务器获得资源。在参考文献[18]中，为无线传感器网络设计了一个可扩展的物联网安全管理架构，并将其性能与现有的管理进行了比较

物联网的解决方案。在参考文献[19]中，提出了一个基于比特币的权利管理系统，它允许用户发布和转让访问策略。其优点是，访问策略将对所有用户开放，并预先阻止任何一方否认策略的真实性。参考文献[20]将区块链和RBAC结合起来，实现了基于以太坊的跨组织RBAC，允许小型组织参与，因此用户可以完全控制他们的角色。在参考文献[21]中，提出了公平访问隐私保护和权利管理框架，该框架使用智能合约以细粒度的方式管理访问策略，并检查策略重用。在参考文献[22]中，提出了一个名为EduRSS的方案。它使用基于Ethereum的区块链来存储和共享教育记录。参考文献[23]提出了一个新颖的基于信任的推荐方案（TBRS），以确保车辆CPS网络中的安全和实时数据传输。在参考文献[24]中，提出了一个基于超图的区块链模型。该模型利用超图结构对存储节点进行排序，将整个网络的数据存储转变为本地网络存储，减少了存储消耗，提高了安全水平。

III. 初步了解

A. Hyperledger fabric

以比特币为代表的数字货币取得了巨大成功，吸引了全世界对区块链技术的关注。然而，这种公有链也有很多缺点，列举如下。

1) 交易吞吐量低。每秒只能接受大约7个交易。

2) 交易确认时间长。每笔交易需要大约1小时才能最终确认。

3) 浪费资源。PoW机制消耗了大量的计算资源和电力。

4) 一致性问题。区块链很容易形成一个分支。当一个分支形成后，只有最长的那条链生效，其他链上的所有交易都是无效的。

5) 隐私问题。由于比特币的账本是公开的，所以交易中没有隐私。

为了解决这些问题，Linux基金会在2015年推出了Hyperledger项目，构建了一个企业区块链开发平台。作为其中的一个项目，Hyperledger Fabric采用模块化结构，提供可扩展的组件，包括加密、认证、共识算法、智能合约、数据存储和其他服务。

Hyperledger Fabric的所有程序都在docker容器中运行。容器提供了一个沙盒环境，它将应用程序与物理资源分开，并将容器相互隔离，以确保应用的安全性。Hyperledger Fabric是一种联盟链，其中所有节点都需要经过授权才能加入区块链网络。在此基础上，Fabric提供了一个基于Kafka消息队列的共识机制，可以在大的规模的应用场景。Hyperledger Fabric克服了公有链的这些缺点。

1) 主要组件。CA、客户、对等人、订购者

*CA*对成员节点的数字证书进行统一管理，并生成或取消成员的身份证书。

*客户端用于与*对等节点互动，同时操作区块链系统。该操作分为两类。第一类是管理类，主要用于管理节点，包括启动、停止、配置节点等。另一类是链码类，主要用于链码的生命周期管理，包括链码的安装、实例化、升级和执行等。客户端一般为命令行客户端或由SDK开发的应用程序。

*对等体*是分布式系统中的一个平等的节点，这个组件存储区块链的账本和链码。应用程序连接到对等体，通过调用链码查询或更新账本。有两种类型的对等体：认可者和承诺者。认可者节点负责验证、模拟和认可交易。承诺者负责验证交易的合法性，以及更新区块链和账本状态。

*订购者*负责接受对等节点发送的交易，根据特定的规则对交易进行分类，将交易按照一定的顺序打包成区块，并将其发送给对等节点。对等节点用新的区块更新本地账本，并最终达成共识。

2) 频道

大多数企业应用的一个重要要求是数据隐私和保密性。The Fabric设计了一个通道系统来隔离不同组织的区块链数据。在每个通道中，都有一个独立的私人账本和一个区块链。因此，Fabric是一个具有多通道、多账本和多区块链的系统。

3) 账本

Fabric的数据以分布式账本的形式存储，该账本中的数据项以键值对的形式存储。所有的键值对构成了账本的状态，这被称为 "世界状态"，如图1所示。

4) 恰恰相反

Fabric中的智能合约被称为Chaincode。Chaincode是一个用golang编写的程序（支持其他编程语言，如Java），实现预定义的接口。交易可以由Chaincode生成，这是外界与区块链系统互动的唯一方式。通过提交或评估交易，外部人员可以改变或读取状态数据库（SDB）的数据，而交易将被写入Fabric的分类账。业务逻辑可以通过编写Chaincode来实现。

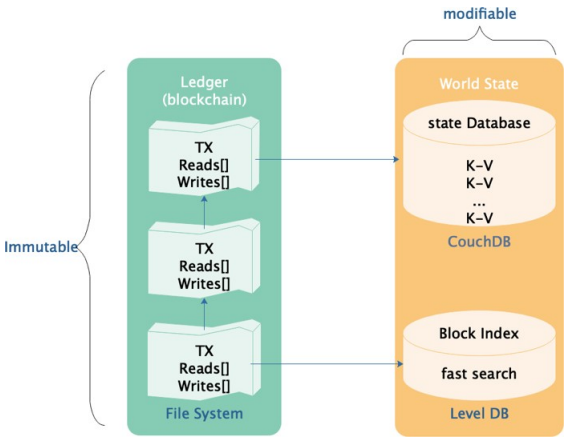


图1.Hyperledger Fabric中的分类账结构。

因此，开发者可以编写不同的链码来实现不同的应用。

B. ABAC模型

基于属性的访问控制（ABAC）是一种访问控制技术，它将主体、客体、许可和环境的属性考虑在内。它通过判断请求者的请求是否包含正确的属性来决定是否授予其访问权。由于主体和客体的属性是单独定义的，ABAC可以有效地将策略管理和访问控制分开。该策略可以根据实际情况进行修改，如增加或减少其属性，以达到可扩展的目的。此外，实体的属性可以从不同的角度来定义，以实现精细化的访问控制。属性是ABAC的核心，它可以定义为一个有四个元素的集合。$A\ S,\ O,\ P,\ E$.每个字段的含义解释如下。

A代表属性，$A = \{name : value\}$。一个属性的值有一个键名。

S代表主体的属性，即发起访问请求的实体的身份和特征，如一个人的身份证、年龄、姓名、职位等。

O代表对象的属性，即访问资源的属性，如资源类型、服务IP地址、网络协议等。

P代表权限的属性，指主体对客体的操作，如读、写、执行等。

E代表环境属性，即访问请求产生时的环境信息，如时间、地点等。

*ABACR（基于属性的访问控制请求）*被定义为。$ABACR = AS \wedge AO \wedge AP \wedge AE$，它是一个包含上述四个属性的集合。它代表AS(Attributes of Subject)的AP（Attributes of Operation）。

表1.资源URL的例子。

URL	意义
https://www.xxx.com/live/test_video.m3u8	重置HLS视频的URL来源
rtmp://www.xxx.com/live/test_video.flv	RTMP视频的URL重新来源
tcp://www.xxx.com/mqtt/test_topic	实时数据URL与主题为 "test_topic"

在AE(Attributes of Environment)下的AO(Attributes of Object)。

*ABACP（基于属性的访问控制策略）*被定义为。$ABACP = AS \vee AO \vee AP \vee AE$，代表主体对客体的访问控制规则。它表达了受保护资源访问所需的属性集。

IV. 系统模型和设计

A. 资源和政策模式

物联网设备产生的数据有很多种，其中大部分是非结构化数据[25]。例如，摄像头可以捕捉现实世界的图像并产生图片或视频数据，麦克风可以捕捉外部声音并产生音频数据。传感器可以捕捉物理信号，如温度、湿度、光线，并将其转换成数字信号数据。这些数据大多是非结构化的，所以不能直接存储在关系型数据库中。而且由于它们都是实时数据，需要及时推送给授权用户。一般来说，语音和视频数据都是流式数据。设备采集的数据经过编码后，通过WiFi或4G推送到云服务器。最后，生成一个资源URL。用户可以根据视频传输协议，如HLS和RTMP，通过URL提取流媒体数据。对于传感器数据，设备通过基于MQTT的[26]服务或其他协议将数据发送至一个主题。客户端获得授权后，订阅相应的主题（可以用URL表示），服务器将该主题下的消息推送给客户端。此外，这类数据主要用于控制物联网设备执行绑定、解除绑定、打开、关闭、调整等操作。一般来说，客户端可以通过基于HTTP（s）的restful API向服务器发送请求。服务器验证权限后，可以通过MQTT或其他协议将控制信号发回给设备。表1显示了几种不同类型资源的URL格式。总而言之，本文定义了一个设备资源模型。$\{设备\} \rightarrow \{resource\} \rightarrow \{url\}$。主体（用户）对客体（再源）的访问权由访问策略定义。用户不是直接从设备上请求资源，而是根据URL从区块链系统中获得资源数据，并带有许可核查。

设备资源和用户之间的联系如图2所示。简要的工作流程显示在图中的1-5个序号中。

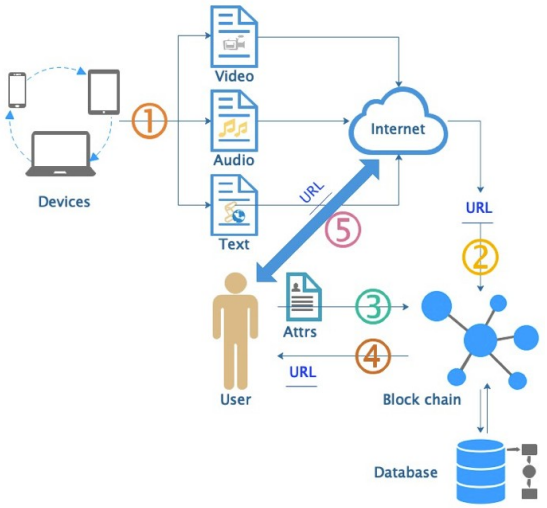


图2.用户和资源之间的联系。

1) 设备将资源分发到互联网，并生成资源的URL。
2) 设备将资源URL保存到区块链系统。
3) 用户通过属性请求区块链系统，以获得授权。
4) 区块链向授权用户发送URL。
5) 用户根据URL在互联网上下载或提取资源数据。

结合ABAC模型和物联网设备产生的数据特点，设备访问控制策略模型定义如下。

$$p=\{as, ao, ap, ae\} \tag{1}$$

$$AS = \{userId, role, group\} \tag{2}$$

$$AO = \{deviceId, MAC\} \tag{3}$$

$$fAP = \begin{cases} 1、允许 \\ 0, \text{deney} \end{cases} \tag{4}$$

$$AE = \{创建时间，\ 结束时间，\ 允许的IP\} \tag{5}$$

P（政策）。它代表归属访问控制的政策。这个集合包含四个元素。AS, AO, AP, 和AE。

AS（主体的属性）。它代表了一个主体（用户）的属性，包括三种类型：userID（唯一标识用户）、role（用户角色）和group（用户组）。

AO（对象的属性）。它代表一个对象（资源）的属性，包括设备ID或设备的MAC地址。在这个模型中，我们不把资源的URL直接看作是一个属性。相反，我们使用唯一的标识作为设备的属性。因为在现实中，设备的网络是可变的，设备产生的数据是动态的。我们假设一个设备在系统中的功能是单一的，每个ID或MAC在同一时间只能对应一个资源URL。

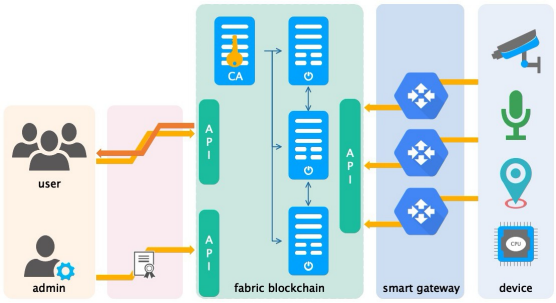


图3.织物-iot的结构。

AP(Attribute of Permission)。它表示用户是否可以访问资源。值1代表"允许"，2代表"拒绝"。当AP被初始化时，默认值为1。管理员可以根据情况将AP的值设置为0来撤销访问授权。

AE（环境属性）。它表示访问控制所需的环境属性。AE有三种属性：时间、结束时间和允许的IP。时间代表策略的创建时间。结束时间代表策略的到期时间。当当前时间晚于结束时间时，该策略将无效。允许的IP是为了防止网段外的IP地址访问系统。

B. 系统结构

Fabric-iot是一个基于区块链的物联网访问控制系统，由四个部分组成：用户、区块链、智能网关和设备，如图3所示。

用户。这个系统将用户分为两种类型：管理员和普通用户。

管理员负责管理区块链系统和维护智能网关的程序。管理员需要提供一个证书来访问区块链系统。允许的具体操作如下。

1) 添加新的智能合约。管理员可以部署新的智能通过API在区块链上的合同。

2) 升级合约。管理员可以将新的智能合约上传到节点并安装，而旧的合约将被升级到新的版本。

普通用户，也就是设备的所有者，通过向区块链系统发送基于属性的授权请求来获得资源URL。

区块链。它是系统的核心。所有节点在加入区块链系统之前都需要获得CA认证。区块链是基于Hyperledger Fabric开发的，它通过智能合约实现访问控制。区块链系统为用户和智能网关的访问暴露了API。它主要实现以下三个功能。

1) 设备资源URL数据存储。
2) 基于属性的用户权限管理。
3) 对用户访问资源进行认证。

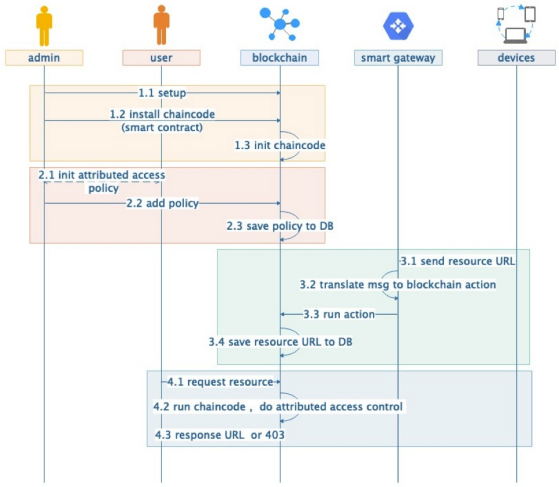


图4.织物-iot的工作流程。

表2.符号描述。

命名	意义
CA	证书颁发机构
资格证书	证书文件
忏悔	节点的配置文件
$F(x)..$	源代码中定义的功能
CC	Hyperledger Fabric中的Chaincode
交流	访问合同
个人电脑	政策合同
直流电	设备合同
图片	Docker图像
容器	Docker 容器
TX	区块链中的交易
AS, AO, AE, AP	主体、客体、许可和环境的属性。运作
账本	Hyperledger Fabric中的Ledger
SDB	Hyperledger Fabric中的状态数据库
AAA	智能网关
文学士	区块链行动
Cli	区块链系统客户端
ABACP	基于属性的访问控制政策

智能网关。作为设备和区块链系统之间的桥梁，它可以接收来自设备的信息，并将其包含的URL放到区块链上，避免了设备直接访问给区块链系统带来的压力。

物联网**设备**。作为最大的群体，物联网设备通常不具备强大的计算能力，也没有足够的存储空间。
，和耐用的电池。因此，不可能直接将物联网设备部署为区块链的对等节点。物联网设备有一个独特的MAC地址或产品ID，它可以与其他设备区分开来。一般来说，这些设备可能同时属于一些用户或群体。每当设备产生一个新的资源时，一个消息包含

资源的URL被发送至智能网关。在此
系统中，MQTT协议被用作消息传输。
协议。

C. 工作流程

如图4所示，整个系统的工作流程主要包括四个部分。本节将详细介绍每个部分的中间步骤。使用的符号在表2中描述。

Part1

区块链网络初始化和链码安装是系统的基础过程。这些操作需要管理员在内部网中工作。流程1主要包括三个步骤。

步骤1 在建立Hyperledger Fabric网络之前，我们需要为所有成员创建证书，如对等节点、订购者节点、通道、用户等。所有证书都由CA生成。

$$CA \rightarrow \{Cert_{peer}, Cert_{order}, Certchannel, Cert_{user}\} \quad (6)$$

同伴节点和订单节点在docker容器中运行。在运行之前，需要将证书打包到他们的docker镜像中。

$$Build(conf, Cert) \xrightarrow{build} Image \xrightarrow{run} Container \quad (7)$$

在所有对等节点和订购者节点设置成功后，我们开始创建通道。每个通道都加入了一个独立的区块链和账本。

$$\{区块链，分类帐\}。\xrightarrow{加入} 通道 \quad (8)$$

步骤2 到目前为止，一个基本的Hyperledger Fabric网络已经建立。为了构建应用程序，我们需要设计chaincode。我们的chaincode的源代码是用Golang写的。

$$代码(F(x)..) \rightarrow CC \quad (9)$$

管理员使用Hyperledger Fabric SDK或客户端来安装CC（Chaincode）。所有CC将被安装到对等节点。

$$安装(CC) \xrightarrow{SDK/客户端} 同行 \quad (10)$$

Step3

一旦Chaincode安装完毕，需要对其进行初始化。Invoke函数被用来初始化chaincode。每一个实例化的Chaincode都将作为一个批注保存在容器中。

$$\underset{SDK/客户端}{调用(初始)} \text{ - - - - } \rightarrow \underset{同行}{} \tag{11}$$

Part2

制定访问控制策略并将其保存到区块链系统。这个过程需要用户和管理员一起工作，事先决定和定制访问策略，并由管理员将其上传到区块链系统。

步骤1

管理员和用户共同制定访问策略，这些策略是根据主体（用户）、对象（设备资源）、操作和环境的属性来定义的。

$$决定(AS, AO, AE, AP) \rightarrow ABACP \tag{12}$$

第2步

在访问策略被定义后，管理员将其上传到区块链网络。

$$上传(ABACP) \rightarrow 合同 \tag{13}$$

Step3

管理员连接到区块链，通过运行PolicyContract来添加、修改和删除政策。价值

这篇文章已被接受发表在本杂志的未来一期中，但尚未完全编辑。内容在最终出版前可能会有变化。引用信息。DOI 10.1109/ACCESS.2020.2968492, IEEE Access

在SDB中保存该政策的信息，并将行动记录写入分类账。

$$PolicyContract(ABACP) \rightarrow \{Ledger, SDB\} \quad (14)$$

第三部分

设备向智能网关报告资源URL，之后，智能网关将其上传至区块链系统。

步骤1 设备产生的信息包括设备ID和URL，并通过MQTT

$$\{deviceId, URL\}。\rightarrow Msg \xrightarrow{MQTT} SG \quad (15)$$

步骤2

智能网关解析消息，并为区块链生成一个动作。

$$Translate(Msg) \rightarrow BA \quad (16)$$

Step3 智能网关连接到区块链客户端来运行动作。

$$运行(BA) \xrightarrow{在} 设备合同 \quad (17)$$

步骤4

区块链通过调用DC的功能来保存设备资源的URL。

$$DeviceContract(deviceId, URL) \rightarrow \{Ledger, SDB \quad (18)$$

第四部分

基于贡献而获取资源的过程是系统的核心。它有以下三个步骤。

步骤1 用户发起基于属性的请求。

$$userId \rightarrow Request\{AS \wedge AO \wedge AP\} \quad (19)$$

步骤2 在收到请求后调用AC的功能。

$$AccessContract(Request) \rightarrow \begin{cases} 1, \text{OK} \\ 0, \text{禁止} \end{cases} \quad (20)$$

Step3

如果验证通过，区块链系统通过调用DC中的函数查询URL并返回给用户。如果验证失败，403错误（403代表HTTP状态代码中的禁止）将被返回给用户。

$$结果= \begin{cases} 1, \xrightarrow{DC} URL \\ 0, \rightarrow 403 \end{cases} \quad (21)$$

D. 智能合约设计。

智能合约是访问控制实现的核心。在这个系统中，有三种类型的智能合约。政策合约（PC），设备合约（DC）和访问控制（AC）。

政策合同：它提供了以下方法来操作ABACP。

Auth()。管理员为用户定义ABACP，并向区块链系统发送添加ABACP的请求。一个ABACPR（基于属性的访问控制策略请求）的例子显示在表3。管理员用PC节点的公钥对数据进行加密，然后用私钥签署该请求。PC调用Auth()，用其公钥验证管理员的身份，并用自己的私钥解密数据。

表3.ABACPR的一个例子。

ABACPR.json
"*Action*" : "*add*", "*Data*" : {{ "*AS*" : { "*userId*" : "10001", "*role*" : "*manager*", "*group*" : "g1", "*AO*" : "*deviceId*" : { "B230011001xxx01", "*MAC*" : "48 : e2 。 *xx* : *xx* : 5*f* : *f* 9", "*AP*" : 1, "*AE*" : "*createTime*" : "1572607208", "*endTime*" : "1575199208", "*allowedIP* : "10.10.100.∗ /10.10.255.∗ "}}}

将其发送到智能网关。

算法1 PolicyContract.CheckPolicy()。检查ABAC政策在将其放入DB之前

输入。ABACP

输出。真或假

```
1: < AS, AO, AE, AP ← >ABACP
2: IsOK = T rue
3: 为AS中的项目做。
4 : 如果项目/< userId, role, group >，那么 5 : IsOK = False
6: 结束 如果
7: 结束
8: for item in AO do:
9: 如果项目/< deviceId, MAC > 那么
10: IsOK = False
11: 结束 如果
12: 结束
13: 如果V al(AE)!= 1 或() 14: IsOK = False
15: end if
16: for item in AP do:
17 : 如果项目/<创建时间，结束时间，允许的IP>，那么
18 : IsOK = False
19: 结束 如果
20: 结束
21: 返回IsOK
```

CheckPolicy()。如算法1所示。PC需要检查ABACP的有效性。一个合法的ABACP需要包含上述四个属性，而且每个属性的类型也需要满足要求。

AddPolicy()。如算法2所示。在ABACP被CheckPolicy()验证为合法后，PC调用AddPolicy()将ABACP添加到SDB中，同时，所有的行动记录将被写入账本。

UpdatePolicy()。在某些情况下，管理员需要修改ABACP。函数UpdatePolicy()实现了ABACP的互操作。更新SDB的脸，更新的操作记录也将被写入区块链。UpdatePolicy()是

类似于AddPolicy()，它也调用应用接口的put方法来覆盖旧值。

算法2 PolicyContract.AddPolicy()。将ABAC策略添加到区块链

输入。ABACP

输出。错误或无效

```
1: @实现SmartContract接口
2: APIstub ChaincodeStubInvoke() 3:
if CheckPolicy(ABACP ) == False 4:
return Error( t BadP olicy t)
5: 结束 如果
6: IdSha256(ABACP.AS + ABACP.AO)
7: err.APIstub.PutState(Id, ABACP )
8: 如果err! = null，那么
9 : 返回Error(err.T ext)。
```

10: 结束 如果

10.1109/ACCESS.2020.2968492, IEEE Access

11: renturn *null*

算法3 PolicyContract.DeletePolicy()。删除ABAC策略从区块链

输入。*AS, AO*

输出。*错误信息*

```
1:@implement SmartContract Interface
2:                          ← APIstubCh
aincodeStubInvoke()
3: IdSha256(AS + AO)
4:      ← errAPIstub.GetState(Id)
5: 如果err! = null，那么
6：返回Error(err.T ext)。
7: 结束 如果
8: APIstub.DelState(Id)
9: if err! = null then
10：返回Error(err.T ext)。
11: 结束 如果
12: renturn null
```

DeletePolicy()。ABACP有一个过期时间，它可以由管理员取消。有两种情况下会发生删除。一种是当管理员通过调用这个函数主动删除一个策略时发生。另一种情况发生在Check- Access()方法执行时，如果属性"endTime "过期，那么它将在PC中调用这个函数来删除相关策略。如算法3所示。

QueryPolicy()。它实现了数据库查询的接口，提供了一个函数来获取其他链码的ABACP。我们选择CouchDB作为SDB。虽然它是一个键值文档数据库，但CouchDB支持与mongoDB类似的复杂查询。在这种情况下，QueryPolicy()支持通过AS或AO查询ABACP。

设备合同：DC主要负责将设备的资源URL存储到SDB中。DC有2个输入参数*DeviceId* 、*URL*。提供的功能如下。

AddURL()。它使用DeviceId作为一个键，URL作为一个值存储在SDB中。

GetURL()。它根据DeviceId从SDB查询相应的URL值。

访问合同：它验证用户的ABACR是否符合ABAC策略。与PC一样，请求数据由用户的私钥签名，之后，AC通过用户的公钥验证签名以检查用户的身份。提供的方法如下。

Auth()。与同名的PC方法类似，它用用户的公钥验证请求，并检查其身份的真实性。

GetAttrs()。它在验证特征后解析属性数据字段。只有部分ABAC属性被包含在ABACR中：*AS, AO* ，而AE需要被AC去掉终端。最后，这些属性被合并为*AS, AO, AE*。

CheckAccess()。它是实现访问控管理的核心函数，如算法4所示。首先，它通过GetAttrs()获取属性设置。其次，它调用PC的QueryPolicy()方法，根据AS和AO查询相应的ABACP。如果返回的结果是空的，这意味着没有政策支持该请求，它将直接返回一个403错误（表示没有权限）。如果返回的结果不是空的，它

算法4 AccessContract.CheckAccess()。检查用户的访问权限

输入。*ABAC_Request*

输出。*URLorError*

```
1: < AuS, AuO, AuE  ← >GetAttrs(ABAC_Request)
2: P =< p 1, p2 , ...P n> ← PC.QueryPolicy(A uS, A uO)
3：如果P ==
Null，那么4：返回E
rror(403)。
5: endif
6：对于< P 1, P 2, . . . P >中的
P，做7：< . . .P n>做7 ← < . .
., ApP, ApE                >P t
8: 如果V alue(ApP ) == deney then
9:      继续
10: 如果AuEqApE == then
11:      继续
12:          ← URLDC.GetURL(AuO)
13: 结束
14: 如果URL! = Null
那么 15: 返回URL
16：否则
17：返回Error(403)。
18: 结束 如果
```

表4.硬件和软件环境。

硬件设施	
CPU	i7 7500u 2.9GHz, i7 8700k 3.7GHz
记忆	8G,8G
硬盘	256G，1T
软件	
操作系统	Mac OS 10.14.6, Deepin Linux 15.11
装载机	v19.03.2
docker-compose	v1.24.1
结点	v12.12.0
golang	v1.12.9
Hyperledger fabric	v1.4.3

表示将获得一个或多个ABACP。第三，开始逐一判断请求的AE是否与ABACP的AE匹配，AP的值是否为1（代表允许）。如果所有属性都符合政策，则验证通过。最后，它调用DC的GetURL()函数来获取资源的URL并返回给用户，否则将返回403错误。

V. 实验和比较

本节介绍了实验过程和结果比较，以证明我们提出的fabric-iot的功能和性能。在第一部分，我们列出了本实验中使用的硬件和软件。在第二部分，我们展示了建立系统的过程和基于系统的访问控制的实现。在最后一部分，我们给出了性能测试、比较实验和结果分析。

fabric-iot项目的源代码在GitHub上是开源的：https://github.com/newham/fabric-iot。

A. 环境

本文的实验是在两台PC上进行的。硬件和软件环境列于表4。

B. 系统构建过程和实现

10.1109/ACCESS.2020.2968492, IEEE Access

实验分为三个部分。第一部分主要介绍 fabric-iot 的结构，以及初始化配置和启动步骤。第二部分

表5.节点的Docker图像。

节点名称	描述	数量
织物-iot/couchdb	数据库节点	4
织物-iot/ca	CA结点	2
织物-iot/peer	同伴节点	4
织物-iot/orderer	订货人节点	1
hyperledger/fabric-tools	超级账本的工具	1
织物-iot/chaincode/PC	政策合约节点	4
织物-iot/chaincode/DC	设备合同节点	4
织物-iot/chaincode/AC	访问合同节点	4

介绍了Chaincode的安装过程。第三部分介绍了如何通过调用三种智能合约（PC、AC和DC）实现基于ABAC的物联网资源访问控制方法。

1) 织物-iot的结构和初始化

Fabric-iot由8种docker节点组成，如表5所示。系统初始化的步骤如下。

第1步：使用Hyperledger加密工具为节点（peer、orderer等）生成根证书和秘钥对。

第二步：将这些证书和密匙对移到指定的目录中，该目录将被CA的docker镜像挂载，在容器运行时生效。其他节点可以用他们的签名向CA验证他们的身份。

Step3：使用configtxgen工具生成一个创世区块，用来打包交易包含节点和通道的配置。当fabric-iot启动后，创世区块被写入区块链，确保整个系统中每个节点的身份信息不能被篡改。之后，其他节点的镜像会根据docker compose的初始化配置来启动。当所有容器运行成功后，对等节点将被添加到一个通道中。

2) Chaincode的安装和升级

在网络成功建立后，我们开始安装Chaincodes。Chaincodes是通过执行hyperledger客户端的指令来安装的。具体步骤如下。

第1步：将chaincodes的源代码复制到客户端节点挂载的目录中。

第二步：运行命令将链码打包到通道中的一个对等节点。

第三步：将每个编译好的链码传送到其他对等节点并将其实例化。每个链码的副本被保存到一个单独的容器中作为背书。

升级过程与安装类似。然而，只有第一次安装chaincode的节点会立即升级到新版本，其他对等节点只有在产生交易时才会同步升级。

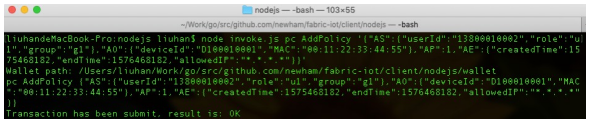


图5.在PC中调用AddPolicy()的结果。

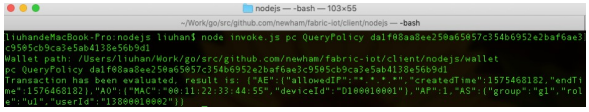


图6.在PC中调用QueryPolicy()的结果。

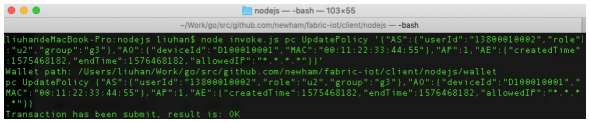


图7.在PC中调用UpdatePolicy()的结果。

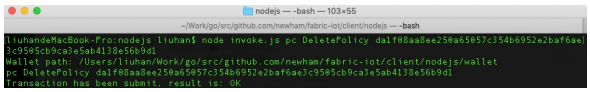


图8.在PC中调用DeletePolicy()的结果。

3) 实施ABAC

在Hyperledger Fabric平台上，有一些方法可以调用Chaincode。外部人员可以使用客户端或SDK（支持Java、golang、node）在Hyperledger Fabric平台上调用Chaincode。

Fabric-iot使用node SDK编写的客户端来调用Chaincode。其步骤如下。

第1步：CA节点为客户生成密匙对，并保存在用户的钱包中。

第二步：管理员运行一个客户端连接到对等节点，以提交或评估（相应的写和读操作）一个交易。

第三步：对等节点在订购者节点的服务下，通过与其他对等节点达成共识来查询或更新SDB。

为了添加ABAC策略，PC的AddPolicy()方法被调用，如图5所示。

为了验证策略是否被成功添加，我们调用PC.QueryPolicy()方法进行查询。查询中使用的密钥是根据公式得到的：$\text{sha256}(\textit{UserId}, \textit{DeviceId}) \rightarrow \textit{key}$。图6中显示了一个查询。

如果真实环境中的属性发生变化或需要更新策略，可以调用PC.UpdatePolicy()函数，如图7所示。删除操作可以通过PC.DeletePolicy()方法实现，如图8所示。

如图9所示，设备可以通过调用DC.AddUrl()来报告资源地址，该URL通过DeviceId或MAC地址与策略相关。

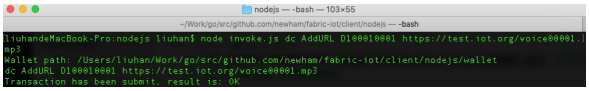


图9.在DC中调用AddURL()的结果。

图10.在DC中调用GetURL()的结果。

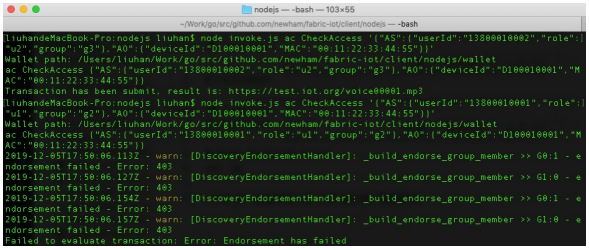


图11.在AC中调用CheckAccess()的结果。

可以通过调用DC.GetURL()来查询URL，如图10所示。

在收到用户的请求后，AC调用Check-Access()方法。它首先调用PC，根据AO和AS查询相关策略，然后检查AE和AP属性是否满足条件。如果条件满足，则证明ABAC的属性约束得到满足。最后，调用DC的GetURL()方法来获取设备资源URL。如果属性条件不满足，将返回一个错误，如图11所示。

C. 结果和比较

为了测试fabric-iot系统的性能，我们设计了两组比较实验，模拟多线程客户端对系统的并发访问。在第一组实验中，我们统计了不同数量的并发请求下PC、AC和DC的处理时间。虚拟客户的数量被设定为50、100、200、500、1000。统计结果显示在图12-17。从这些数字可以看出。

1) 写（如"添加"、"更新"）操作比读（如"获取"、"查询"）操作花费更长的时间。

2) 系统的吞吐量随着请求数的增加而增长。当吞吐量达到一定值时，它趋于稳定。而随着客户数量的进一步增加，吞吐量没有明显的下降趋势。

在第二组实验中，我们根据Hyperledger Fabric中的共识机制原理，使用MQTT网状队列对交易进行排序，模拟fabric-iot的共识算法。我们用golang语言实现了PoW共识算法，并设置了合理的难度以适应实验环境。我们测试了

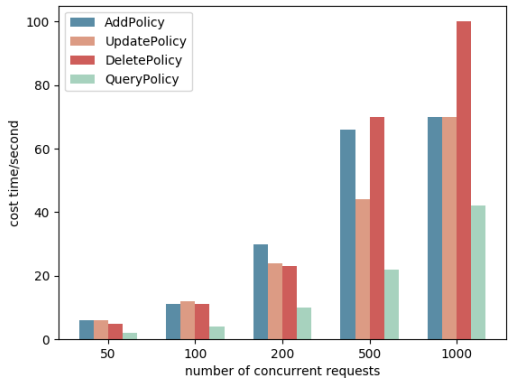


图12.在不同数量的并发请求下，PC的成本时间。

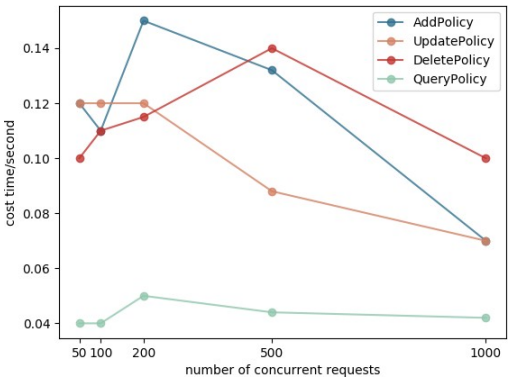


图13.在不同数量的并发请求下，PC的平均成本时间的趋势。

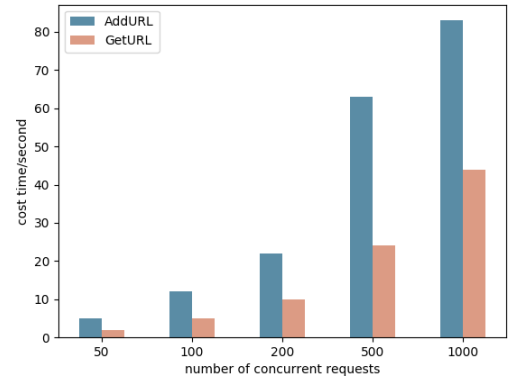


图14.不同数量的并发请求下DC的成本时间。

通过比较不同节点数下fabric-iot和PoW共识机制的成本时间，研究分布式系统的数据一致性效率。实验中的节点数设置为5到100。实验结果如图18所示。从图中可以看出，在

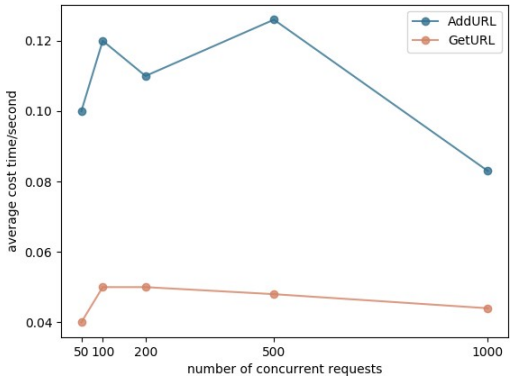


图15.在不同数量的并发请求下，DC的平均成本时间的趋势。

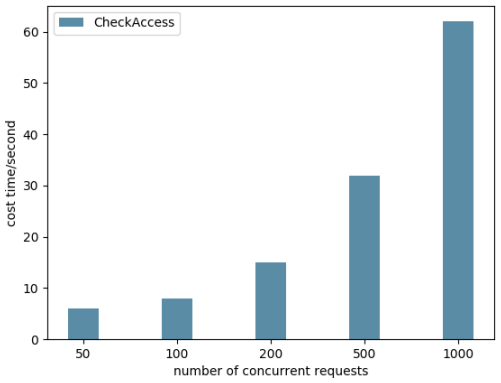


图16.不同数量的并发请求下AC的成本时间。

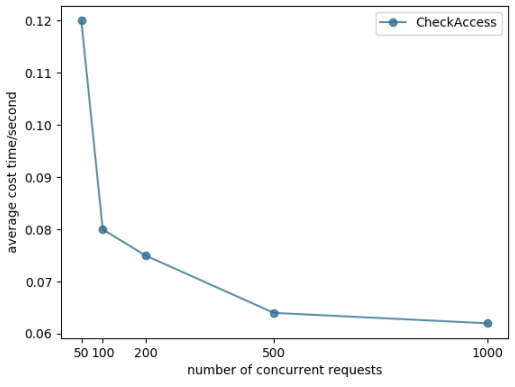


图17.在不同数量的并发请求下，AC的平均成本时间的趋势。

确保PoW安全的难度，在fabric-iot中达成共识的成本时间远远低于PoW。

以上两组实验可以证明，fabric-iot可以在大规模再生产中保持高吞吐量。

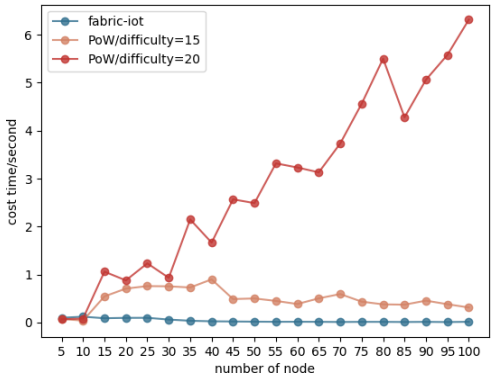


图18.fabric-iot和PoW之间的共识速度比较。

quest环境，并能在分布式系统中有效达成共识，保证数据的一致性。

VI. 结论和进一步工作

本文将区块链技术与ABAC模型相结合，利用区块链技术的去中心化、防篡改和可追溯性等优势，解决了传统的基于中心化设计的访问控制方法难以满足物联网访问控制要求的问题。首先，根据物联网设备的实际生产数据，我们提出了一个设备权限模型。其次，根据ABAC模型，我们实现了ABAC策略管理，并通过实施智能合约应用确保设备资源的访问安全。此外，我们设计并实现了一个基于Hyperledger Fabric的开源访问控制系统，名为fabric-iot。该系统采用分布式架构，可以为物理网络提供细粒度和动态的访问控制管理。最后，详细描述了区块链网络构建、链码安装、智能合约调用等步骤，并通过实验展示了令人信服的结果。最重要的是，本文为其他研究人员开展相关研究提供了实用参考。

未来的工作可以在以下方面进行改进。

1) 本文的实验是在两台PC上进行的。在未来，我们考虑使用集群或边缘计算服务来部署，并进一步验证该系统的分布式性能。

2) 在未来，可以使用更多的物理设备来测试系统的可靠性和吞吐量。

3) 未来的研究可以尝试提高fabric-iot的可扩展性，并支持更多的物联网应用整合。

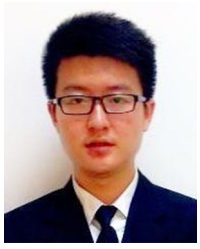
参考文献

[1] J.Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things:架构、使能技术、安全和隐私以及应用"，IEEE物联网杂志，第1-1页。

[2] K.Chopra, K. Gupta, and A. Lambora, "未来的互联网。The internet of things-a literature review," in 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing（COMITCon）, Feb 2019, pp.
[3] J.Wang, H. Wang, H. Zhang, and N. Cao, "Trust and attribute-based dynamic access control model for internet of things," in 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery（CyberC）, Oct 2017, pp.
[4] V.C. Hu, D. R. Kuhn, and D. F. Ferraiolo, "Attribute-based access control," Computer, vol. 48, no. 2, pp.
[5] N.聪聪，"比特币--开源P2P货币。访问，" 2019年11月。[在线]。Available: https://bitcoin.org/en/
[6] E.Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A.De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich等人, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in Proceedings of the Thirteenth EuroSys Conference.ACM，2018，第30页。
[7] W.Liang, M. Tang, J. Long, X. Peng, J. Xu, and K.-C.Li, "A secure fabric blockchain-based data transmission technique for industrial internet of things," IEEE Transactions on Industrial Informatics, 2019.
[8] O.Novo, "区块链与iot相遇。An architecture for scalable access management in iot," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1184-1195, 2018.
[9] Y.Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the internet of things," IEEE Internet of Things Journal, vol. 6, no. 2, pp.1594-1605, 2018.
[10] G. Papadodimas, G. Palaiokrasas, A. Litke, and T. Varvarigou, "Implemen- tation of smart contracts for blockchain based iot applications," in 2018 9th International Conference on the Network of the Future（NOF）.IEEE，2018，pp.60-67。
[11] K.Košťál, P. Helebrandt, M. Belluš, M. Ries, and I. Kotuliak, "Management and monitoring of iot devices using blockchain, " Sensors, vol. 19, no.4, p. 856, 2019.
[12] J.Yang, S. He, Y. Xu, L. Chen, and J. Ren, "A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks, " Sensors, vol. 19, no.4, p. 970, 2019.
[13] S.Ding, J. Cao, C. Li, K. Fan, and H. Li, "A novel attribute-based access control scheme using blockchain for iot, " IEEE Access, vol. 7, pp. 38 431- 38 441, 2019.
[14] M.Ma, G. Shi, and F. Li, "Private-oriented blockchain-based distributed key management architecture for hierarchical access control in the iot scenario," IEEE Access, vol. 7, pp. 34 045-34 059, 2019.
[15] Z.Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," IEEE Internet of Things Journal, pp.1-1.
[16] M.Cui, D. Han, and J. Wang, "An efficient and safe road condition monitoring authentication scheme based on fog computing," IEEE Internet of Things Journal, vol. 6, no.5, pp. 9076-9084, 2019.
[17] J.Pan, J. Wang, A. Hester, I. Alqerm, Y. Liu, and Y. Zhao, "边缘链。基于区块链和智能合约的边缘-iot框架和原型，"IEEE物联网杂志，2018。
[18] O.Novo, "Scalable access management in iot using blockchain: a performance evaluation," IEEE Internet of Things Journal, 2018.
[19] D.D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control," 2017.
[20] J.P. Cruz, Y. Kaji, and N. Yanai, "Rbac-sc:使用智能合约的基于角色的访问控制，"IEEE Access，第6卷，第12240-12251页，2018。
[21] A.Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "Fairaccess: a new blockchain-based access control framework for the internet of things," Security and Communication Networks, vol. 9, no. 18, pp.5943-5964, 2016。
[22] H.Li and D. Han, "Edurss:A blockchain-based educational records secure storage and sharing scheme," IEEE Access, vol. 7, pp. 179 273-179 289, 2019.
[23] W.Liang, J. Long, T.-H.Weng, X. Chen, K.-C.Li, and A. Y. Zomaya, "Tbrs:A trust based recommendation scheme for vehicular cps network," Future Generation Computer Systems, vol. 92, pp.383-398, 2019.
[24] C.Qu, M. Tao, and R. Yuan, "A hypergraph-based blockchain model and application in internet of things-enabled smart homes, " Sensors, vol. 18, no. 9, p. 2784, 2018.
[25] W.梁，K.-C.Li, J. Long, X. Kui, and A. Y. Zomaya, "An industrial network intrusion detection algorithm based on multi-feature data clustering optimization model, " IEEE Transactions on Industrial Informatics, 2019.
[26] N.Tantitharanukul, K. Osathanunkul, K. Hantrakul, P. Pramokchon, and P.Khoenkaw, "Mqtt-topics management system for sharing of open data," in 2017 International Conference on Digital Arts, Media and Technology（ICDAMT）, March 2017, pp.

刘 瀚
在上海海事大学获得硕士学位，目前正在该校攻读博士学位。他的主要研究兴趣包括大数据、云计算、分布式计算、云安全、机器学习、物联网和区块链。

韩 德 智 在
华中科技大学获得博士学位。他目前是上海海事大学计算机科学与工程学院的教授。他的研究兴趣包括云计算、移动网络、无线通信和云安全。



李 敦
在澳门科技大学获得硕士学位，目前在上海海事大学攻读博士学位。他的主要研究兴趣包括智能金融、大数据、机器学习、物联网和区块链。