基于 Client Puzzle 的公有链接入控制模型

巫岱玥 李 强 余 祥 黄海军
(国防科技大学 合肥 230037)

摘 要 公有链无集中控制节点、去中心化和允许任意节点加入的特点使其拥有高效、低成本和高数据安全性的优势,但由于其允许任意节点接入,因此将增加公有链网络的脆弱性。基于 Client Puzzle,提出一种节点接入控制模型 CPACM(Client Puzzle based Access Control Model),使新节点接入公有链前利用算力进行工作量证明,完成工作量证明后才可接入公有链。该模型在维持公有链去中心化的同时,增加了接入控制。实验证明,该模型在不影响诚实节点加入的情况下,能以较高成功率限制低诚意节点和低算力节点的加入,并且能防止节点间的伙同,防范了恶意行为,提高了公有链网络的安全性。

关键词 公有链,区块链,接入控制,计算谜题,工作量证明
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.04.021

Client Puzzle Based Access Control Model in Public Blockchain

WU Dai-yue LI Qiang YU Xiang HUANG Hai-jun
(National University of Defense Technology, Hefei 230037, China)

Abstract Public blockchain characterizes non-centralization and decentration and allows any node to join, and thus possesses the advantages of high efficiency, low cost and high data security. However, that it allows any node to join increases the vulnerability of blockchain network. This paper proposed a Client Puzzle based access control model named CPACM. In this model, new nodes have to make use of computing power to conduct proof of work before joining the network. Only the successful nodes could join the public blockchain network. This model adopts access control when keeping public blockchain decentralizing. Experimental results show that the proposed model can restrict the low computing-power nodes and unfaithful nodes to access with high probability when not affecting the faithful nodes to access and prevent nodes collusion, thus preventing the malicious behaviors and improving the security of public blockchain.

Keywords Public blockchain, Blockchain, Access control, Client Puzzle, Proof of work

1 引言

区块链按照其开放程度不同可以划分为公共区块链(即公有链)、联盟区块链(即联盟链)、私有区块链(即私有链)3种。其中,联盟链和私有链具有严格的访问控制机制,只有相应权限的用户才能加入到区块链网络,对区块链数据进行读写,通常认为联盟链和私有链是“部分去中心化”的;公有链则没有访问控制,用户的加入完全没有门槛且无需审核,任何人都可以加入公有链网络,进行区块链数据读取、数据生成、共识达成等。公有链被认为是“完全去中心化”的,解决了现有中心化架构普遍存在的成本高、效率低、存储不安全和单点失效的问题。

公有链保证了数据安全但未保证公有链网络的安全[1]。公有链允许任意节点不需花费开销就能加入网络,其中不乏恶意节点、低诚意节点和低算力节点,这些节点的加入会增加公有链网络的脆弱性,如增加了网络中发生 Sybil 攻击、DDoS 攻击和 Eclipse 的可能性[2]。例如 2015 年,比特币网络遭受 Sybil 攻击,大量虚假节点加入并伪造成全节点,造成了网络性能的大幅下降;2017 年 3 月比特币无限(bitcoin unlimited)遭遇大规模 DDoS 攻击,短时间内大量节点下线,对网络的健壮性造成了严重的威胁。

目前对公有链的保护措施分为两种,即对节点进行访问控制与事后恶意节点检测。针对任意节点都可以加入区块链网络,文献[3]提出所有节点必须经过认证中心 CA 的认证才能接入网络,没有授权的节点无法接入网络,不能获得交易信息和区块信息,并将此方案应用到超级账本(Hyperledger)中[14]。虽然该方案实现了对节点加入的控制,但是基于认证中心的授权机制削减了区块链的去中心化特性,实质上将公有链转变成了私有链。在保证公有链去中心化特性的前提下,文献[4]提出了一种基于行为模式聚类的恶意节点检测方

到稿日期:2018-03-05 返修日期:2018-08-23 本文受国防科技大学科研基金项目(KYJ2017J351)资助。
巫岱玥(1994-),男,硕士生,主要研究方向为区块链;李 强(1962-),男,教授,硕士生导师,主要研究方向为区块链、软件工程、信息安全,E-mail:lychfeei@163.com(通信作者);余 祥(1986-),男,硕士,讲师,主要研究方向为信息安全;黄海军(1975-),男,博士,讲师,主要研究方向为信息安全。

法，能快速定位恶意节点并将其加入黑名单，阻止恶意节点继续进行恶意行为。由于公有链缺少对接入的控制，恶意节点被移除公有链网络后，能够通过更改自身身份标志 ID，反复加入网络，继续进行恶意行为[11]。

现有的两种对公有链的保护措施是矛盾的，文献[3]增加认证中心可以在节点加入前进行审核，但会削减公有链的开放和去中心化特性；若维持公有链开放和去中心化特性，通过其他检测方法检测恶意节点，如文献[4]中的方法，则只能起到事后补救的作用，无法防范恶意行为的发生，且恶意节点在被移除后可以再次加入网络。本文采用 Client Puzzle 方法，将 Client Puzzle[5-7] 运用于节点的加入阶段，建立基于 Client Puzzle 的接入控制模型 CPACM(Client Puzzle based Access Control Model)，链内节点依据交互结果决定请求方是否能够加入网络。该模型不依赖中心节点，使得公有链既维持了去中心化特性又增加了对节点的接入控制。

本文第 2 节描述 Client Puzzle 方法；第 3 节结合公有链特点，构造一种适用于公有链的 Client Puzzle 的方案；第 4 节利用该方案构建 CPACM 模型，以达到对加入节点的接入控制的目的；第 5 节进行实验验证，证明了该接入控制模型的有效性，最后总结全文。

2 Client Puzzle 方法

Client Puzzle 方法指请求方(client)获得响应方提供的服务前，需要请求方进行工作量证明，具体来说响应方会产生难题 puzzle 并发送给请求方，请求方需要解决这个难题并且得到响应方的认可才能获得响应方提供的服务，puzzle 的求解需要请求方付出一定的代价，即工作量，代价以请求方资源(CPU、内存等)的形式支付。一个完整的 Client Puzzle 分为 4 个阶段，分别是 Client Puzzle 的初始化阶段、puzzle 产生阶段、puzzle 求解阶段、puzzle 验证阶段，由 $(Setup, GenPuz, FindSoln, VerSoln)$ 四元组组成：

$CPuz=(Setup, GenPuz, FindSdoln, VerSoln)$

$Setup(1^k)$ 是 CPuz 初始化阶段，Setup 分为 6 个步骤：

1)k 为 CPuz 的安全指数，根据安全指数 k 初始化私钥集合 $sSpace$、CPuz 难度集合 $diffSpace$、字符串集合 $strSpace$、CPuz 难题集合 $puzSpace$、CPuz 答案集合 $solnSpace$，并决定这 5 个集合的元素数目和各集合中元素的长度。k 值等于私钥集合 $sSpace$ 中私钥的长度，该值应设计得足够大以确保私钥的安全性。

2)从私钥集合 $sSpace$ 中选取一个私钥 s，记为 $s \leftarrow sSpace$。

3)从难度集合 $diffSpace$ 中选择一个难度系数 d，记为 $d \leftarrow diffSpace$。

4)从字符串集合 $strSpace$ 中选择一个字符串 str，记为 $str \leftarrow strSpace$。

5)$params \leftarrow (sSpace, puzSpace, solnSpace, diffSpace, \Pi)$，其中 $\Pi \in params$ 是 CPuz 描述信息，如 CPuz 难题的描述、CPuz 的适用范围。

6)return$(params, s, d, str)$。

GenPuz 是 Client Puzzle 生成阶段，此阶段根据 Setup 阶段产生的 s、d、str 生成难题 puzzle，记为：$GenPuz(s \in sSpace, d \in diffSpace, str \in strSpace)$：return$(puzzle \in puzSpace)$。

FindSoln 是 Client Puzzle 解答阶段，此阶段根据 GenPuz 阶段产生的难题 puzzle 生成符合条件的解 $soln$，记为：$FindSoln(puzzle \in puzSpace, t \in N)$：return $soln \in solnSpace$。其中，t 表示解决 CPuz 所需的时间开销。

VerSoln 是 puzzle 解答验证阶段，此阶段验证 FindSoln 阶段产生的解 $soln$ 是否满足 GenPuz 阶段产生的难题 puzzle，记为 $VerSoln(puz \in puzSpace, soln \in solnSpace)$：return TRUE or FALSE。

一种设计正确的 Client Puzzle 方案的难题的解应该能在有限时间内得到，记为：

if$(params, s, d, str) \leftarrow Setup(1^k)$ && $puz \leftarrow GenPuz(s, d, str)$

then $\exists t(t \in \mathbb{N} \rightarrow \Pr(VerSoln(puz, soln) = \text{TRUE} \mid soln \leftarrow FindSoln(puz, t) = 1)$

下面根据定义构造基于强 Hash 碰撞的 Client Puzzle。

令消息认证码 $MAC(\cdot): \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^k$，$\Pi$ 是单向散列函数 $H(\cdot): \{0,1\}^* \rightarrow \{0,1\}^d$。

$Setup(1^k)$：$sSpace \leftarrow \{0,1\}^k$，$diffSpace \leftarrow \{1,2,\cdots,k\}$，$strSpace \leftarrow \{0,1\}^*$，$puzSpace \leftarrow diffSpace \times strSpace \times \{0,1\}^k$，$solnSpace \leftarrow \{0,1\}^k \times \{0,1\}^k$，$s \leftarrow sSpace$，$d \leftarrow diffSpace$，$str \leftarrow strSpace$，$params \leftarrow (sSpace, puzSpace, solnSpace, diffSpace, \Pi)$，return$(params, s, d, str)$；

$GenPuz(s, d, str)$：$\sigma \leftarrow MAC(s, str \| d)$，$puz \leftarrow (d, str, \sigma)$，return puz；

$FindSoln((str, d, \sigma), t)$：find $x_1, x_2 \in \{0,1\}^k$，$H(str \| x_1) = H(str \| x_2)$，return (x_1, x_2)；

$VerSoln((str, d, \sigma), (x_1, x_2))$：if $H(str \| x_1) = H(str \| x_2)$，return TRUE else return FALSE.

此 Client Puzzle 所产生的 puzzle 要求 client 找到一对散列值相同的比特串，具体来说是要求 client 找出两个不同的比特串 x_1 与 x_2，使得 str 拼接 x_1 并进行散列计算得到的散列值与 str 拼接 x_2 并进行散列计算得到的散列值相同。

3 公有链的 Client Puzzle 的构造

公有链为保证其数据一致性和共识机制，引入了分布式算力竞争[15-16]。各节点基于其算力相互竞争解决一个求解复杂但是验证容易的 hash 问题，该 hash 问题可以表述为：通过搜索求解一个适合的随机数使区块头的哈希值小于或等于目标哈希值。公有链网络是依靠算力的网络[13]，区块链的节点对计算能力的要求较高，因此为公有链构造的 Client Puzzle 在强调难题解答需要花费代价的同时，还应强调对节点计算能力的测试，只有满足算力要求的节点才能通过测试。

应用于公有链的 Client Puzzle 应满足如下约束条件：

1)构造和验证 puzzle 所需的代价很低。由于验证节点可能会在一段时间内向多个请求节点发送 Client Puzzle 和验证 Client Puzzle，若生成和验证 Client Puzzle 的代价过大，会导致验证节点的生成和验证的效率下降，严重的会导致请求节点停止服务[10]。

2)请求节点不能预先计算出 Client Puzzle 的答案。

3)请求节点进行 puzzle 求解时,认证节点无需存储此 puzzle 的相关信息。验证节点可能会为众多请求节点生成对应的 puzzle,如果存储每一条 Client Puzzle 将占用请求节点的大量资源。

4)认证节点为请求节点生成的 puzzle 只能用于此请求节点的验证。

5)puzzle 难度能被调节。

6)puzzle 只能依赖算力求解。

基于上述约束条件设计 Client Puzzle。

设单向散列函数为 $H(\cdot):\{0,1\}^* \to \{0,1\}^k$,签名函数为 $F_S(\cdot):\{0,1\}^* \to \{0,1\}^k$,$\Pi$ 是单向散列函数 $H(\cdot)$,根据 Client Puzzle 定义:

$Setup(1^k)$:$sSpace \leftarrow \{0,1\}^k$,$diffSpace \leftarrow \{1,2,\cdots,k\}$,$strSpace \leftarrow \{0,1\}^*$,$puzSpace \leftarrow diffSpace \times strSpace \times \{0,1\}^k$,$solnSpace \leftarrow \{0,1\}^*$,$params \leftarrow (sSpace, puzSpace, solnSpace, diffSpace, \Pi)$,$s \leftarrow sSpace$,$d \leftarrow diffSpace$,$str \leftarrow strSpace$,return $(params,s,d,str)$;

$GenPuz(s,d,str)$:$x \leftarrow H(d,str)$,$\sigma = F_S(x)$,$puzzle \leftarrow (d,str,\sigma)$,return $puzzle$;

$FindSoln((d,str,\sigma),t)$:$y \leftarrow \{0\}^d$,$while\{soln \leftarrow \{0,1\}^*$,$y' \leftarrow H(soln,str)$,if $y = y'.left(d)$ then return $soln\}$;

$VerSoln((d',str',\sigma'),soln)$:$y \leftarrow \{0\}^d$,$y' \leftarrow H(soln,str')$,if $y = y'.left(d)$ then return TRUE,else return FALSE.

此 Client Puzzle 要求请求方解决一个部分散列碰撞问题,由于理想的 Hash 函数具有单向性,且请求方无法通过散列值倒推出散列前的信息,请求方想要找到满足条件的比特串只依靠暴力穷举,因此对请求方的计算能力有较高的要求。Client Puzzle 可以根据公有链对节点的算力要求来调节难度系数 d,以筛选出满足公有链算力要求的节点,因此该方案不仅满足 puzzle 的解答较为耗费资源而验证十分容易的要求,还能筛选出满足公有链算力需求的节点。

请求方根据响应方给定 puzzle,构造特定数据以计算出符合条件的散列值,具体来说请求方需要构造比特串 $soln$,使 $soln$ 与 str 组成的比特串经过散列运算后满足前 d 位比特串为 0,其中 d 是难度系数。通过简单改变 str 的内容可以生成不同的 puzzle。通过少量 hash 运算可以验证 puzzle 的解是否正确,满足条件 1)。

定理 1　若 puzzle 的难度系数为 d,则节点求解单个 puzzle 所需的计算量为 2^d。

证明:考虑一般情况,令 $t = t_1 + t_2 + \cdots + t_n$,其中 t_i 代表解决第 i 个 puzzle 所花费的计算次数,则:

$$P(t_i) = (1-\frac{1}{2^d})^{t_i-1} \cdot \frac{1}{2^d}$$

每个 puzzle 的解答是独立的,因此 $\prod_1^n P(t_i) = P(\prod_1^n t_i)$,解决 n 个 puzzle 的概率为:

$$P(t) = \prod_1^n P(t_i) = \prod_1^n (1-\frac{1}{2^d})^{t_i-1} \cdot \frac{1}{2^{nd}} = (1-\frac{1}{2^d})^{t-n} \cdot \frac{1}{2^{nd}}$$

又 $t_1, t_2, \cdots, t_n$,存在 $\binom{t-1}{n-1}$ 种可能方案,因此,需要计算的次数为 $\zeta_{k,d,n}(t) = \sum_n^t \binom{t-1}{n-1} \cdot \frac{1}{2^{nd}} \cdot (1-\frac{1}{2^d})^{i-n}$。

当 $n=1$ 时,$\zeta_{k,d,1}(t) = \sum_1^\infty i \cdot \frac{1}{2^d} \cdot (1-\frac{1}{2^d})^{i-1} = \frac{1}{2^d} \cdot \sum_1^\infty i \cdot (1-\frac{1}{2^d})^{i-1} = \frac{1}{2^d} \lim_{i\to\infty} \frac{i \cdot (1-\frac{1}{2^d})^{i-1} \cdot (-\frac{1}{2^d}) - (1-\frac{1}{2^d})^i + 1}{\frac{1}{2^{2d}}} = 2^d$。证毕。

推理 1　若 puzzle 的难度系数为 d,无法构造出计算量低于 2^d 的 Client Puzzle。

证明:Client Puzzle 由交互双方的唯一身份标识与随机数组成。理想情况下,认为 hash 函数是随机预言机,即请求双方无法由 hash 值倒推出原像,由定理 1 可知,交互双方欲构造出特定的原像(包含唯一身份标识与随机数)使得 hash 值满足条件的概率 $P = \frac{1}{2^d}$,因此无法通过构造特殊的 Client Puzzle 来减少计算量。

由推论 1 可知,交互双方无法通过构造特殊 Client Puzzle 来减少 puzzle 求解的计算量。

puzzle 在 GenPuz 阶段生成,puzzle 的生成依赖于 str 的生成,str 的生成由双方协商产生,请求节点在 GenPuz 阶段完成之前无法得到完整的 puzzle,也无法在 puzzle 生成前预先计算出 puzzle 的解,因此满足条件 2)。

GenPuz 阶段生成的 puz 包含该 puzzle 的全部信息和全部信息的数字签名,验证节点收到来自请求节点的解答后,通过验证数字签名来确定该 puzzle 是否是自己之前产生的,而无需将 Client Puzzle 信息存于请求节点本地,因此满足条件 3)。

双方协商后产生的 str 中包含请求节点和验证节点的身份信息,明确了 Client Puzzle 的交互双方,一旦修改身份信息,将无法通过数字签名的验证,因此满足条件 4)。

puzzle 的求解难度与难度系数 d 呈正相关,改变难度系数 d 可以调整 puzzle 难度,因此满足条件 5)。

puzzle 的求解需要穷举比特串并不断进行 hash 运算直到找到满足条件的比特串,puzzle 的求解只依赖节点算力,因此满足条件 6)。

4　基于 Client Puzzle 的接入控制模型

现有的公有链网络方案中,任何节点都可以不费任何代价加入网络,其中包括低诚意节点、低算力节点,这些节点在网络中的比例的提升会增加公有链网络的脆弱性。

节点 i 的计算开销 C_i 描述为算力 P_i 与时间开销 t 的乘积:

$$C_i = P_i * t \tag{1}$$

定义 1　请求节点 i 的算力 P_i 与理论上该难度系数 d 的 puzzle 被求解所需算力 P_d 的比值为算力率 R_i:

$$R_i = \frac{P_i}{P_d} \tag{2}$$

算力率 R_i 量化了节点算力与求解 puzzle 所需算力的关

系，反映了节点 i 解决 puzzle 的能力。$R_i>1$ 表示节点算力大于理论上求解此难度 puzzle 所需要的算力。

定义 2 Client Puzzle 中开销最大的原子操作为此 Client Puzzle 的基本操作。

定义 3 理论上求解 puzzle 所执行基本操作的开销为该 Client Puzzle 的强度 T。

定义 4 请求节点 i 为解决此次 puzzle 所付出的开销 C_i 与该难度 d 的 puzzle 的强度 T_d 的比值为诚意率 H_i：

$$H_i=\frac{C_i}{T_d} \tag{3}$$

诚意率 H_i 反映了请求节点为加入网络的诚意，$H_i>1$ 表示节点 i 为解决某难度 puzzle 所付出的开销大于该难度 puzzle 的强度。特别地，当 $R_i=1$ 且 $H_i=1$ 时，有 $T_d=C_i$。

诚实节点指节点算力不小于求解 puzzle 所需算力 P_d 且开销不小于 puzzle 的强度 T_d 的节点，根据定义 1 和定义 4 可知诚实节点的算力率 $R_i\geqslant 1$，诚意率 $H_i\geqslant 1$。

低算力节点指算力远低于诚实节点求解 puzzle 所需平均算力的节点，即 $P_i\ll P_d$，$R_i\ll 1$。

无算力节点指没有计算能力的节点，即 $R_i=0$。

低诚意节点指节点算力足够但是对 puzzle 的求解时间远小于标准求解时间的节点，这类节点为求解 puzzle 付出的开销远低于该难度的 Client Puzzle 强度，即 $R_i\geqslant 1$，$t\ll t_d$，$H_i\ll 1$。

无诚意节点指不愿为 Client Puzzle 的求解耗时的 $t=0$ 的节点，无诚意节点的诚意率 $H_i=0$。

恶意节点是指加入公有链之后进行恶意行为的节点。通常这类节点是以诚实节点的形态加入，节点的恶意行为并不在加入阶段发生，而是在节点加入网络之后发生。恶意行为发生后，节点被移除公有链网络，同时向全网广播恶意节点的身份标识，将其加入黑名单列表并记录于区块链账本。本文只研究恶意节点在加入阶段的行为与开销。

构建基于 Client Puzzle 的公有链接入控制模型 CPACM。CPACM 模型主要用于解决以下问题：

1）筛选出诚实节点，限制非诚实节点的加入。

模型要求待加入节点解答 Client Puzzle，而解答 Client Puzzle 需要消耗计算资源，从而以提高加入门槛的方式筛选出诚实节点，限制非诚实节点。

2）筛选出满足算力要求的节点。

公有链网络是高度依赖算力的网络，高算力节点可以增加网络的健壮性并且自身可以获得高收益，低算力节点对网络没有贡献且自身无法获得收益。通过调整 Client Puzzle 难度可以筛选出满足算力要求的节点。

3）提高恶意节点加入网络的代价。

4）防止去中心化网络中节点间的伙同。

接入控制模型序列图如图 1 所示，模型的参与方分为请求节点、响应节点、记账节点和全网节点。请求节点首先向响应节点发出接入请求，响应方收到请求方的请求后向请求节点发出工作量请求证明，请求节点根据要求完成证明，随后回应响应节点，经响应节点确认后，响应节点将此次 Client Puzzle 及答案在公有链网络中广播，同交易数据的传播机制相同，直到下一个区块的记账节点接收到这些信息[12]。记账节点验证请求节点对 Client Puzzle 的解答是否有效，然后将交易与有效的 Client Puzzle 一同打包生成区块，并向全网广播，分发到全网节点。当区块接上区块链后，代表全网大部分节点都认可了请求节点，请求节点被允许加入网络，同时全网任意节点都可以对请求节点的解答进行验证[12]。

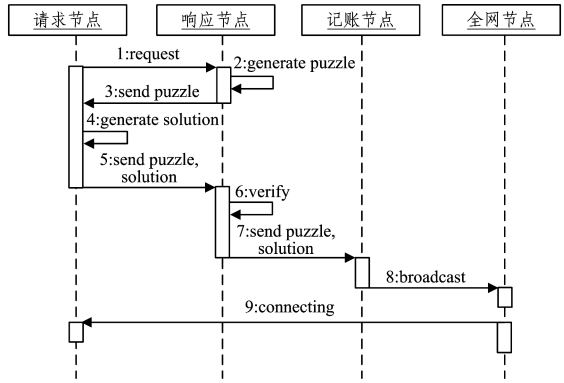


图 1 CPACM 模型序列图

Fig. 1 Sequence diagram of CPACM

在一般的公有链系统中，新节点接入公有链时，公有链系统会将一组长期稳定运行的“种子节点”推荐给新节点建立连接，或者推荐至少一个节点连接到新节点[8]。由于系统推荐的节点均是网络内表现良好且稳定的节点，因此在本模型中这些节点作为响应方更加具有可信度。同时，受公有链共识机制影响，新节点是否能够加入取决于链内大部分节点的态度而不是某个单一节点，即少数服从多数，由此维持了公有链的去中心化，同时也防止了节点间的伙同。节点伙同原指 P2P 网络中节点间相互串通以达到某一目的，在 CPACM 模型中是指公有链网络内节点与请求节点串通，使请求节点无需消耗算力或使用虚假 Client Puzzle 解答就可接入网络。这样的节点在网络中所占的比例称为伙同率。与区块链需要全网大部分节点认可不同，P2P 网络内节点加入无需得到全网节点认可，只需要响应节点认可，因此可能存在伙同行为。若响应节点无法处理节点请求，请求方可以向其他节点发起接入请求，从而避免了单点失效。该算法的具体描述如算法 1 所示。

算法 1 *ClientPuzzle*()

```
描述:Client Puzzle 过程的算法描述
输出:Client Puzzle 完成状态(false 表示未完成,true 表示完成)
n_c←C.g_rand()
C send (id_c, n_c) to S
puz←S.GenPuz(id_c, n_c)
if puz==NULL
then return stop
else
    S send puz to C
    soln←C.FindSolution(puz)
If soln==NULL
then goto step 2
else
    C send puz and soln to S
    if S.VerifySolution(puzzle,soln)==true
    then return true
    else return false
```

1）请求节点准备接入公有链网络前，首先与公有链中一

个节点建立通信,请求节点 C 将自己的 id 记为 id_c,并与生成的随机数 n_c 一同发送给节点 S,等待响应节点 S 的响应。

2)puzzle 生成算法 GenPuz 是确定性算法,如算法 2 所示。响应方每隔一个固定时间使用 $g_{rand}()$ 函数随机生成 n_s 并存储本次的随机数,当响应方收到节点 C 的 id_c 和随机数 n_c 后检查此 id 是否合法(是否在黑名单内),若合法则将节点 C 的 id_c、节点 S 的 id_s、节点 C 的随机数 n_c 和此时间段节点 S 的随机数 n_s 依次拼接生成 str。随机数 n_c 和 n_s 使得同一组交互双方多次交互也能每次产生不同的 str,从而保证了请求方每次收到的 puzzle 都是唯一的。为了防止 d 和 str 在传输过程中被篡改,将难度系数 d 与 str 拼接并使用哈希函数生成消息摘要 x,随后节点 S 使用私钥 s 对摘要 x 进行数字签名得到 σ。最后将难度系数 d、字符串 str 和数字签名 σ 作为 puzzle 发送给节点 C。

算法 2　$GenPuz(id_c, n_c)$

```
描述:生成难题 puzzle
输入:响应发起方的身份 idc 和随机数 nc
输出:生成特定的 puzzle
puz←NULL
if idc∈Blacklist
then
return NULL
else
    if ns
      ns←grand()
      str=(idc ‖ ids ‖ nc ‖ ns)
      x=H(d ‖ str)
      σ=Fs(x)
      puz←(d,str,σ)
    end if
return puz
```

3)puzzle 解答算法 FindSolution 是非确定性算法,如算法 3 所示。由于传输过程中数据明文可能被篡改,我们将节点 C 收到节点 S 发送的 puzzle 记为 puz′,$puz' \rightarrow (d', str', \sigma)$。使用节点 S 的公钥对 σ 进行解密得到 x,令 $x' = H(d' \| str')$,判断 x 与 x' 是否一致。若一致,表示传输过程中 str 与 d 没有被篡改,$str = str'$,$d = d'$;若不一致,表明 puzzle 被篡改,节点 C 将重新生成随机数 n_c 并再次向节点 S 发送交互请求。

从 puz' 中得到 $puzzle$ 难度值 d 和字符串 str,节点 C 随机生成不大于 2^k 长度的比特串,记为 $soln$。令 $y = H(soln \| str')$,要求解的难题转化成 $y.Left(d) =_{(left\ d)} 0^k$,其中 0^k 表示长度为 k 的全零比特串,等式的含义是 y 的前 d 个比特均为 0。即 puzzle 要求节点 C 找出一个比特串,使其 Hash 值左侧 d 比特全为零。对于理想的单向散列函数,找到该难题的解只能通过暴力穷举来得到,理论上需要探测约 2^d 次才能得到符合条件的 $soln$ 比特串[9]。请求方经过若干次计算后得到满足 puzzle 的比特串 $soln$ 后,节点 C 将解 $soln$ 同难题 puz' 一同发送给节点 S。

算法 3　FindSolution(puz)

```
描述:通过暴力穷举寻找符合条件的比特串
输入:难题 puzzle
输出:符合 puzzle 条件的解 soln
set x←Fs^-1(σ)
x′←H(d′ ‖ str′)
if x==x′
  while true
      soln←grand()
      Y←H(str′ ‖ soln′)
      if(y. Left(d)=(left d)0^k)
      then return soln
      end if
else return NULL
end if
```

4)puzzle 验证算法 VerifySolution 是确定性算法,如算法 4 所示。节点 S 收到 puz' 与 $soln$ 比特串后,首先进行身份校验,判断 str' 中的 id_c 是否为当前交互节点的 id,若身份校验成功,再检查随机数是否是此刻本地保存的随机数。若两随机数不一致,说明随机数被篡改或已过期,将终止对话;若随机数有效,再验证 puz' 的合法性,即判断 puz' 是否被篡改。如果 puz' 不合法,终止对话;如果 puz' 合法,将比特串 $soln$ 与比特串 str' 拼接进行哈希运算得到 k 位比特串 Y,若比特串 Y 满足 d 个前导零,说明节点 C 成功解决了节点 S 提出的 Client Puzzle,并通过节点 S 的验证。

算法 4　VerifySolution(puzzle,soln)

```
描述:验证 soln 是否是难题 puzzle 的解
输入:答案 soln 与难题 puzzle
输出:如果验证成功,则返回 true,否则返回 false
if id′!=idc
  return false
if ns′! =ns
  return false
set x←Fs^-1(σ)
x′←H(d′ ‖ str′)
if x==x′
then Y←H(soln ‖ str′)
  If Y. Left(d′)==(left d′)0^k
    return true
  else return false
else return fasle
end if
```

定理 2　当网络中诚实节点数目大于 50%时,请求节点与响应节点将无法伙同。

证明:请求节点与响应节点进行串通有两种情况:1)请求节点可以直接与响应节点建立连接而不进行 puzzle 解答;2)响应节点与请求节点伪造 Client Puzzle 解答信息。

如果响应节点与请求节点直接相连,区块链网络中其余节点在账本中没有找到与请求节点相关的 Client Puzzle 信息,其余节点将不会响应请求节点的任何请求,没有解决Client Puzzle 的节点会被孤立。

如果响应节点为请求节点构造虚假 Client Puzzle 解答信息并向全网广播,记账节点会验证 Client Puzzle 的正确性,虚假 Client Puzzle 解答将无法通过记账节点的验证;若记账节点与请求节点伙同,记账节点将虚假 Client Puzzle 解答信息打包生成区块并广播到全网,将无法通过公有链的共识,区块被视作无效区块。

定理 3　若 puzzle 的难度系数为 d,则恶意节点 n 次加入

网络所需开销为 $n \cdot 2^d$。

证明：节点因恶意行为被移除网络后，其身份标识被加入区块链账本中，且无法被修改或删除，若节点试图使用此身份再次加入网络，响应节点将查询到此身份在黑名单中，从而拒绝此节点的请求。节点想要再次加入网络只能通过改变其身份标识，这将使节点重新进行 Client Puzzle 运算。若恶意节点被移除 $n-1$ 次后，在第 n 次加入网络时，总共需要改变 n 次身份标识，所需开销为 $n \cdot 2^d$。

puzzle 3 个阶段的开销如表 1 所列。puzzle 的生成阶段只执行 1 次 Hash 操作，其计算开销为 1 次 Hash；puzzle 解答过程平均需要 2^d+1 次 Hash 操作，因此 Client Puzzle 的强度为 2^d+1；验证 *soln* 正确性需要 2 次 Hash 操作，puzzle 验证阶段的计算开销为 2 次 Hash。puzzle 的生成和验证操作在响应方进行，响应方完成一次请求方的接入请求仅执行 3 次 Hash，puzzle 的解答过程在请求方进行，开销为 2^d+1 次 Hash。通过分析 3 个阶段的开销可以看出 puzzle 生成阶段和验证阶段的开销远远小于 puzzle 求解阶段，证明了本文提出的 Client Puzzle 方案满足在响应端容易产生 puzzle 和验证 puzzle，请求端需要进行大量计算才能解出 puzzle 的条件。

表 1 3 个阶段的计算开销

Table 1 Computing cost of three steps

阶段	节点	计算开销
puzzle 生成	响应方	1 Hash
puzzle 求解	请求方	2^d+1 Hash
puzzle 验证	响应方	2 Hash

5 分析验证

实验主要考察 CPACM 接入控制模型中节点的接入开销、各种节点的接入控制能力和抗节点伙同能力。实验方案如表 2 所列，在 CPACM 公有链环境下进行接入开销实验，分析节点的接入开销；由于目前尚无其他公有链接入控制模型，我们在 CPACM 公有链与基本公有链环境下进行对比实验，评价两者对节点接入控制的性能差异；在 CPACM 公有链与 CPACM 的 P2P 网络进行对比实验，评价两者在抗节点伙同方面的性能差异。

表 2 实验方案

Table 2 Experimental schemes

实验类型	实验环境	环境构成
接入开销实验	CPACM 公有链	基本公有链+CPACM
接入控制实验	CPACM 公有链	基本公有链+CPACM
	基本公有链	基本公有链
节点伙同实验	CPACM 公有链	基本公有链+CPACM
	CPACM P2 网络	P2P+CPACM

实验环境由待加入节点和公有链构成，具体如图 2 所示，部署测试网络 Testnet 作为本次实验的基本公有链，通过向网络中节点增加接入控制模块实现 CPACM 公有链，根据实验需要更改待加入节点的性质。测试网络 Testnet 是专门用于测试的区块链，它允许程序开发者或者区块链使用者进行功能性测试或试验性操作，具备公有链的一切特性。接入控制模块使用 Python 实现，当配置有接入控制模块的主机收到来自链外节点的接入请求后，会根据请求内容，生成特定的 puzzle 并发送给请求节点。

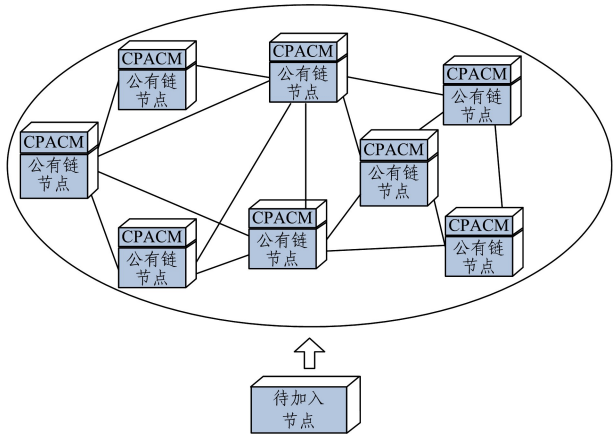


图 2 实验环境

Fig. 2 Experimental environment

5.1 接入开销分析

本节设计实验并记录了节点的实际计算量以分析不同难度下强度 T_d 与实际计算量的关系。实验模拟请求节点加入公有链的过程，请求节点首先向公有链内任意节点发出加入请求，在请求节点接入网络后，记录下本次求解 puzzle 所用开销。

实验参数如表 3 所列，为使节点实验开销贴近实际应用时对节点加入网络的开销要求，共记录了难度系数 d 为 9～21 时计算 puzzle 答案所用基本操作的次数，在每个难度系数下均计算 1000 次并记录下解决该次 Client Puzzle 所用的基本操作次数，每个难度系数下得到 1000 组数据，实验总共记录了 13000 组数据。为了更好地反映每个难度下数据的分散状况，将实验的 13000 次数据统计生成盒须图，并为测试数据添加回归线，如图 3 所示。

表 3 实验参数

Table 3 Experimental parameters

参数	参数属性	描述
Hash 函数	sha256	采用 sha256 作为 hash 函数
n_s	64 bit	响应节点产生 64 位随机数
n_c	64 bit	请求节点产生 64 位随机数
PK_s	64 bit	响应节点 ID
PK_c	64 bit	请求节点 ID

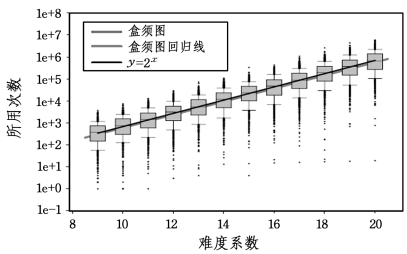


图 3 实验数据盒须图

Fig. 3 Box-plot of experimental data

盒须图常用于统计分析，显示数据到四分位点的分布，突出显示平均值和离群值。盒须图描述了数据中的 5 个统计量：最小值、第一四分位数、中位数、第三四分位数、最大值。由图 3 可以看到，每个 d 都对应一个盒须图，盒须图最下方的数据点的纵坐标值代表该难度系数下求解 Client Puzzle 花费的最少次数，盒须图最上方数据点的纵坐标值代表该难度系数下求解 Client Puzzle 所用的最多次数，盒子的上下两端的

纵坐标表示对应数据的上下四分位数,盒子内部的线段代表该难度系数下求解 Client Puzzle 所用次数的中位数。

上下四分位点间即盒子内部表示数据的集中范围,当难度系数 d 取某值时,求解此难度下的 Client Puzzle 解所需的次数大部分位于上下四分位点间。由于不同 d 值产生的数据数量级差别大,对纵坐标采用对数坐标轴。为反映开销随难度变化的总体趋势,将实验数据进行拟合,由回归线可知实验数据随横坐标呈线性回归,由于 y 轴是对数坐标轴,故实验数据随难度系数 d 的变化呈指数变化。

$y=2^x$ 是 Client Puzzle 的强度线,x 对应的 y 值是 $d=x$ 时 Client Puzzle 的强度。回归线和强度线基本重合,难度为 d 的 Client Puzzle 的强度为 2^d,强度随难度 d 呈指数增加,节点在难度 d 下解答 puzzle 所需的计算量近似等于强度,这意味着节点加入网络的算力代价可以通过 d 被较精确地控制在某一范围内,从而筛选出符合条件的节点。

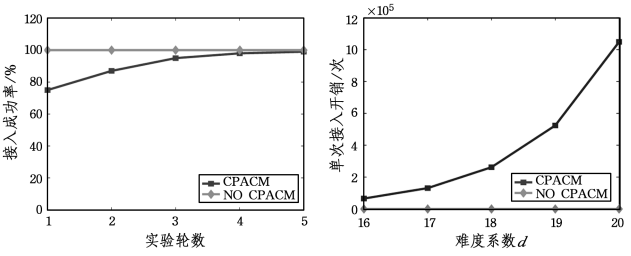
5.2　节点接入实验

为验证 CPACM 模型的有效性,在采用 CPACM 的公有链系统中进行诚实节点、低算力节点和低诚意节点的接入实验,并与没有采用 CPACM 模型的公有链系统进行对比,记录分析 3 种节点的加入情况。

5.2.1　诚实节点接入

分别在采用 CPACM 的公有链和不采用 CPACM 的公有链环境下进行诚实节点的接入实验。

首先对 CPACM 模型的公有链进行实验。在 5.1 节的实验环境下,调整模型难度系数,使 d 分别为 16,17,18,19,20,用 $R_C=1$,$H_C=1$ 的节点对公有链进行接入,每个难度进行 1000 次成功接入。节点首先进行第一轮解答,若解答完成则停止,若节点第一轮解答超时则开始第二轮解答,再次进行接入,以此类推,直到完成解答。记录节点接入网络的成功率并取均值。然后移除实验中的 CPACM 模块,保持实验环境不变再次进行实验,记录节点的成功率。节点接入成功率如图 4 所示。



(a)接入成功率　　(b)单次接入运算次数

图 4　诚实节点接入结果

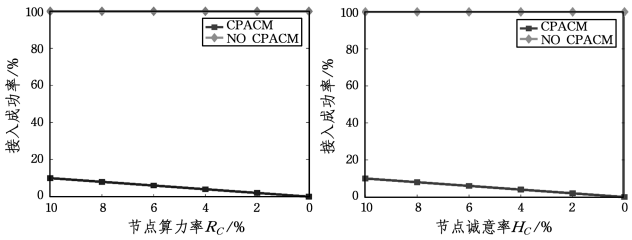
Fig. 4　Accessing results of honest node

由图 4 可知,在 CPACM 模型下,算力率 $R_C=1$,诚意率 $H_C=1$ 的节点第一轮解答后成功接入网络的概率超过 70%,而经过三轮解答后,节点加入公有链的成功率在 95%以上。对于正常接入的节点来说,最多进行三轮 puzzle 求解就能以 95%以上的成功率加入网络,诚实节点加入网络的实际开销基本在$[2^d,3\cdot 2^d]$之间,单次接入的运算次数随难度系数的增加呈指数增长;而没有采用 Client Puzzle 的公有链由于其开放性,允许任何节点的接入,且加入网络也无需任何开销,因此节点加入网络的概率总为 1,开销总为 0。

5.2.2　低算力节点与低诚意节点接入

分别在采用 CPACM 模型的公有链和不采用 CPACM 模型的公有链环境下进行低算力节点的接入实验。

首先对 CPACM 模型的公有链进行实验。在 5.1 节的实验环境下,选取算力率分别为 0,2%,4%,6%,8%,10%的节点作为实验的低算力节点,诚意率为 0,2%,4%,6%,8%,10%的节点作为实验的低诚意节点,分别在难度 $d=16,17,18,19,20$ 下进行实验,共 35 组实验,每组实验进行 1000 次,记录平均接入成功率。然后保持实验环境不变,仅移除 CPACM 模块,再次进行实验并记录接入成功率。节点接入成功率如图 5 所示。



(a)低算力节点接入成功率　　(b)低诚意节点接入成功率

图 5　低算力与低诚意节点接入成功率

Fig. 5　Accessing success rate of low computing-power and unfaithful nodes

由图 5(a)可知,当 $R_C=0$ 时,节点是无算力节点,接入采用本文接入控制模型的公有链的成功率为 0,模型阻止无算力节点的成功率达到 100%,可见没有计算能力的节点是无法加入公有链的。节点接入公有链的成功率随 R_C 的降低而降低,模型阻止低算力节点加入公有链的成功率随节点算力的降低而增加。当低算力节点算力率小于 10%时,模型能够阻止超过 90%的低算力节点的加入,可以看出模型能够有效限制低算力节点的加入,且对算力越低的节点限制效果越好。

由图 5(b)可知,节点加入网络的成功率随诚意率 H_C 的减小而降低,当低诚意节点的诚意率 $H_C\leqslant 10\%$时,模型阻止低诚意节点加入的成功率超过 90%,当 $H_C=0$ 时,节点是无诚意节点,在 5 个难度下节点加入公有链的成功率均为 0。由此证明本接入控制模型可以阻止无诚意节点加入网络,并能阻止大部分低诚意节点加入网络。可以看出,对于采用 CPACM 模型的公有链,无论是低算力节点还是低诚意节点,最终能成功接入的概率都很低,根据式(1)可知其原因在于无论是低算力还是低诚意节点,在规定时间内节点的实际开销远小于解决 Client Puzzle 的应有开销,故无法完成工作量证明。对于没有 CPACM 模型的公有链,由于缺少接入控制,无论节点算力率和诚意率多低,其接入成功率都为 1,因此不能对低算力和低诚意节点起到限制接入的作用。

5.3　节点伙同实验

分别在采用 CPACM 的公有链和采用 CPACM 模型的 P2P 网络环境下进行节点伙同实验。节点加入成功率如图 6 所示。当伙同率为 0 时,公有链网络和 P2P 网络内没有与请求节点伙同的节点,受到接入控制模型的限制,请求节点接入的成功率为 0。随着节点伙同率的增加,P2P 网络的接入成功率呈线性增加,这是因为网络内与请求节点伙同的节点数量增加,响应节点是伙同节点的概率增加;对于公有链,当伙

同节点数量小于节点总数的 1/2 时，受公有链“少数服从多数”的共识机制控制，超过半数的节点不认可请求节点，因此节点的接入成功率仍然为 0。当公有链网络的伙同节点超过半数时，节点接入成功率明显增加。而对于正常网络来说，正常节点的数目应该超过半数，否则对网络的研究是没有意义的，因此当网络中节点伙同率小于 50%时，采用本文提出的接入控制模型的公有链系统可以防止节点伙同。

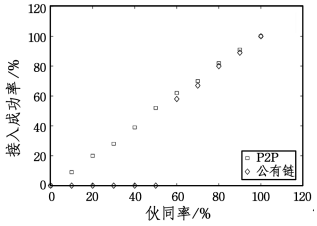


图 6 节点接入成功率

Fig. 6 Accessing success rate of nodes

伙同率与算力开销比的关系如图 7 所示，当节点伙同率为 0 时，算力开销比为 1，这是因为网络内没有与请求节点伙同的节点，请求节点只有进行严格计算才能得到正确解答。随着节点伙同率的增大，P2P 网络环境下节点加入网络的开销逐渐减少，在伙同率达到 100%时，节点接入网络无需开销。在公有链网络内，当节点伙同率低于 50%时，算力开销比仍为 100%，当节点伙同率大于 50%后，算力开销比随伙同率的增加呈线性降低。

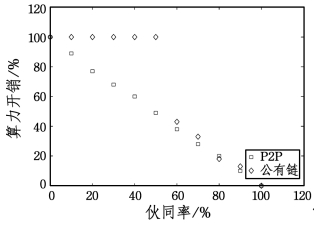


图 7 算力开销

Fig. 7 Computing-power cost

结束语 本文研究了新节点加入公有链的接入控制问题，提出了一种基于 Client Puzzle 的公有链接入控制模型，并对模型的 puzzle 生成、求解和验证 3 个阶段进行了详细的描述。本文提出了算力率与诚意率描述节点特征的方法，使用强度指标描述 Client Puzzle 的开销，对模型中节点在每个阶段的开销进行了理论分析，在不同网络环境下进行了节点接入开销实验、接入对比实验和节点伙同对比实验。实验结果表明，实际开销与理论强度基本一致，模型能在维持公有链开放与去中心化的同时，防止非诚实节点的接入，并有效防止节点伙同，提高了公有链的网络安全性。

鉴于公有链网络随时间的迁移对节点的算力要求会发生改变，节点的加入难度应该随算力的要求而改变。在下一步研究中，将考虑响应节点根据当前公有链网络需求动态调节难度系数，使加入的节点更加精确地满足公有链网络的要求，从而使公有链网络保持良好的性能。

参 考 文 献

[1] LI X, JIANG P, CHEN T, et al. A Survey on the security of blockchain systems[J]. Future Generation Computer Systems, 2017, 9(5): 147-154.

[2] BABAIOFF M, DOBZINSKI S, OREN S, et al. On Bitcoin and red balloons[J]. Acm Sigecom Exchanges, 2011, 10(3): 5-9.

[3] Hyper Ledger. Hyper Ledger architecture working group paper [EB/OL]. [2017-11-10]. https://hyperledeger.org/.

[4] HUANG B, LIU Z, CHEN J, et al. Behavior pattern clustering in blockchain networks[J]. Multimedia Tools & Applications, 2017, 76(19): 1-12.

[5] CHEN L, MORRISSEY P, SMART N P, et al. Security Notions and Generic Constructions for Client Puzzles[C]// International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology. Springer-Verlag, 2009: 505-523.

[6] STEBILA D, KUPPUSAMY L, RANGASAMY J, et al. Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistant Protocols[M]// Topics in Cryptology- CT-RSA 2011. 2011: 284-301.

[7] CHEN R C, GUO W J, TANG L Y, et al. Adaptive Client Puzzle Scheme Against Denial-of-Service Attacks[J]. Journal of Software, 2009, 20(9): 2558-2573.

[8] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system [OL]. http://bitcoin.org/bitcoin.pdf.

[9] WANG Z Y, LI B, ZHANG H G. Research on Security of Hash Funtions[J]. Computer Engineering and Applications, 2005, 41(12): 18-19. (in Chinese)
王张宜，李波，张焕国. Hash 函数的安全性研究[J]. 计算机工程与应用，2005，41(12)：18-19.

[10] ZHANG Z X, DU Y J, LI B, et al. Self-defence model of SIP proxy server for against Dos attack[J]. Journal on Communications, 2009, 30(4): 93-99. (in Chinese)
张兆心，杜跃进，李斌，等. SIP 代理服务器抗拒绝服务攻击自防御模型[J]. 通信学报，2009，30(4)：93-99.

[11] ZHU L H, GAO F, SHEN M, et al. Survey on Privacy Preserving Techniques for Blockchain Technology[J]. Journal of Computer Research and Development, 2017, 54(10): 2170-2186. (in Chinese)
祝烈煌，高峰，沈蒙，等. 区块链隐私保护研究综述[J]. 计算机研究与发展，2017，54(10)：2170-2186.

[12] LIU M D, SHI Y J. Remote Attestation Model Based on Blockchain[J]. Computer Science, 2018, 45(2): 48-52. (in Chinese)
刘明达，拾以娟. 基于区块链的远程证明模型[J]. 计算机科学，2018，45(2)：48-52.

[13] Underwood S. Blockchain beyond bitcoin[M]. ACM, 2016.

[14] LI W, SFORZIN A, FEDOROV S, et al. Towards Scalable and Private Industrial Blockchains[C]// ACM Workshop on Blockchain, Cryptocurrencies and Contracts. ACM, 2017: 9-14.

[15] HE P, YU G, ZHANG Y F, et al. Survey on Blockchain Technology and Its Application Prospect[J]. Computer Science, 2017, 44(4): 1-7. (in Chinese)
何蒲，于戈，张岩峰，等. 区块链技术与应用前瞻综述[J]. 计算机科学，2017，44(4)：1-7.

[16] YUAN Y, WANG F Y. Blockchain: The State of the Art and Future Trends[J]. Acta Automatica Sinica, 2016, 42(4): 481-494. (in Chinese)
袁勇，王飞跃. 区块链技术发展现状与展望[J]. 自动化学报，2016，42(4)：481-494.