工程科学学报
Chinese Journal of Engineering

基于零知识证明的智能合约投票系统设计与实现

殷红建 朱岩 王静 郭光来 陈娥

Design and implementation of a smart-contract voting system based on zero-knowledge proof

YIN Hong-jian, ZHU Yan, WANG Jing, GUO Guang-lai, CHEN E

引用本文:
殷红建, 朱岩, 王静, 郭光来, 陈娥. 基于零知识证明的智能合约投票系统设计与实现[J]. 工程科学学报, 优先发表. doi: 10.13374/j.issn2095-9389.2022.07.07.001
YIN Hong-jian, ZHU Yan, WANG Jing, GUO Guang-lai, CHEN E. Design and implementation of a smart-contract voting system based on zero-knowledge proof[J]. *Chinese Journal of Engineering*, In press. doi: 10.13374/j.issn2095-9389.2022.07.07.001

在线阅读 View online: https://doi.org/10.13374/j.issn2095-9389.2022.07.07.001

工程科学学报, 第 45 卷, 第 X 期: 1−11, 2022 年 X 月
Chinese Journal of Engineering, Vol. 45, No. X: 1−11, X 2022
https://doi.org/10.13374/j.issn2095-9389.2022.07.07.001; http://cje.ustb.edu.cn

基于零知识证明的智能合约投票系统设计与实现

殷红建[1)], 朱　岩[1)], 王　静[2)], 郭光来[1)], 陈　娥[1)]✉

1) 北京科技大学计算机与通信工程学院, 北京 100083　2) 海信集团控股股份有限公司, 青岛 266071
✉通信作者, E-mail: chene@ustb.edu.cn

摘　要　作为一种具有法律约束力的程序, 智能合约为电子投票系统提供了可信执行平台. 然而, 由于合约部署在公开透明的区块链上, 这将为投票内容的有效性与隐私性带来巨大威胁. 为了解决上述问题, 基于交互式零知识证明技术设计了智能合约投票系统. 首先, 提出了一个新的交互式零知识集合成员关系证明协议, 使得投票者在不泄露投票内容的前提下, 完成对投票内容有效性的验证, 从而避免无效选票对投票系统的影响. 其次, 本文通过智能合约规范语言 SPESC 对投票合约进行描述并对投票各个阶段的触发条件进行限定, 通过将合约以 JAR 包形式上传至区块链, 实现智能合约投票系统的部署和自动化执行. 最后, 对智能合约投票系统的性能进行分析, 实验结果表明该系统投票和计票阶段均可高效实施, 为密码协议构造技术与智能合约投票系统的有效结合提供参考.

关键词　零知识证明; 智能合约; 投票系统; 同态加密; 隐私保护
分类号　TP309.2

Design and implementation of a smart-contract voting system based on zero-knowledge proof

YIN Hong-jian[1)], *ZHU Yan*[1)], *WANG Jing*[2)], *GUO Guang-lai*[1)], *CHEN E*[1)]✉

1) School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
2) Hisense Group, Qingdao 266071, China
✉ Corresponding author, E-mail: chene@ustb.edu.cn

ABSTRACT　As a legally binding computer program, smart contracts are stored on the blockchain and can be automatically executed according to the contract terms. These features of smart contracts provide a trusted execution environment for the electronic voting system. However, since the contract is deployed on an open and transparent blockchain, this causes a considerable threat to the validity and privacy of the voting content. However, due to the openness of the blockchain network, any node linked to the network can obtain information concerning contract transactions on the chain without restriction, which greatly threatens the validity and privacy of the voting content. To address this problem, a smart-contract voting system has been designed. First, we construct a new interactive zero-knowledge set membership proof protocol (ZSMPP) based on the discrete logarithm problem. Using ZSMPP in the design of the smart-contract voting system, the voter can verify the voting content validity to the initiator without disclosing the voting content itself to avoid the impact of invalid votes. Moreover, we prove that the proposed protocol is complete and has zero knowledge. Second, we describe the voting contract by the specification language of smart-contract (SPESC) and limit the trigger conditions of stages of the voting system through contract terms. By deploying the voting contract to the blockchain as a JAR file, the proposed smart-contract voting system can be automatically executed in accordance with the predefined contract terms. Additionally, we further introduce the execution process and related algorithms of the four stages of the proposed voting system and show the related execution results in the form of contract

收稿日期: 2022−07−07
基金项目: 国家科技部重点研发计划资助项目(2018YFB1402702); 国家自然科学基金资助项目(61972032)

transactions. Furthermore, we analyzed five security features of the proposed voting protocol. Particularly, the validity of the ballot content is ensured by the zero-knowledge of our protocol, which can prevent invalid votes from affecting the system. The privacy of the ballot ensures that the voting content is undisclosed either in the verification or counting stage. Uniqueness ensures that each voter can only vote once. Supervision-free means that there are no trusted supervisors in the proposed voting protocol. Self-counting indicates that smart-contract programs automatically implement the counting process. Finally, the performance of the proposed smart-contract voting system is analyzed. The experimental results show that both the voting and counting stages of our voting system can be implemented efficiently. Moreover, our smart-contract voting system can provide a reference for effectively combining the cryptographic protocol construction technology and smart-contract voting system.

KEY WORDS　zero-knowledge proof; smart contract; voting system; homomorphic encryption; privacy protection

智能合约作为第二代区块链技术的核心, 其本质是部署在区块链上的一系列可执行数字协议. 智能合约兼顾区块链和合约的双重属性, 具有去中心化、不可篡改、可编程和法律化等特征[1], 已经被广泛应用于金融、数字资产管理等诸多领域[2−5]. 近年来, 随着支持智能合约开发的区块链平台相继提出, 例如以太坊[6]、根链(RootStock)[7]以及 Hyperledger Fabric[8], 开发人员可通过可编程的语言(如 Solidity, Java)在这些平台上实现对智能合约的开发、部署和执行. 然而, 区块链平台的开放性使得部署在区块链上的智能合约是公开透明的, 因此合约交易中的所有数据也都是公开可见的, 这势必引起合约交易中敏感信息的泄露问题. 在电子投票合约中, 由于合约数据涉及用户的隐私信息且投票内容不能公开, 因此合约隐私泄露问题尤为突出.

尽管现有智能合约平台支持密码机制(例如椭圆曲线密码 ECC)可用来保护投票合约中选票内容的隐私, 但在计票阶段仍需对选票解密后再进行统计, 这将为投票内容的隐私带来威胁. 此外, 为了避免无效选票对系统的干扰, 需在投票之前对选票内容的合法性进行验证, 并且验证过程不能泄露投票内容信息. 显然, 现有智能合约中的密码机制不能满足智能合约投票系统对数据隐私及安全性的需求. 因此, 设计新的能够保护交易隐私的智能投票合约系统是十分必要的和迫切的.

1981 年, Chaum[9] 首次提出了安全电子投票系统的概念, 此后涌现出了一大批关于电子投票协议构造以及安全性的研究[10−13]. 目前为止, 尽管人们对其安全性仍存在一定争论[14], 但电子投票系统已在实际环境中得到广泛应用, 甚至被用于国家选举[15]. 随着区块链技术的出现与发展, 基于区块链的电子投票系统被相继开发. 2015 年 Zhao 与 Chan [16] 设计了第一个基于区块链的电子投票方案, 该方案能够在比特币网络中直接运行. 随后, Tarasov 与 Tewari[17] 基于 Zcash 设计了一个新的电子投票协议, 协议中投票结果的正确性由可信任第三方和投票者共同保证. 2017 年, McCorry 等[18] 设计了第一个去中心化的能够实现自动计票功能的投票协议, 并在以太坊平台以合约的形式进行了实现. 然而, 该系统仅支持小规模候选人模式, 而对于大范围投票, 该协议则不再适用. Yu 等[19] 利用环签名技术设计了基于智能合约的可验证投票系统并在 Hyperledger Fabric 平台上进行了部署实现.

此外, 为防止无效选票对投票系统带来的干扰, 需对选票有效性进行验证. 零知识集合成员关系证明协议 (Zero-knowledge set membership proof protocol, ZSMPP)[20] 可为选票有效性验证提供技术支持, 通过零知识证明技术, ZSMPP 协议允许在不泄露选票内容的情况下对公开候选人集验证选票. 2008 年, Camenisch 等[20] 提出了集合成员关系判定问题, 并基于 Strong Diffie-Hellman (SDH) 假设在双线性群下构造了第一个 ZSMPP 协议. 随后, Morais 等[21] 利用数字签名方案[22] 对文献 [20] 的证明阶段进行优化, 使计算开销降低到常数级别. 最近, Yin 等[23] 基于聚合函数构造了同时支持属于和不属于关系的 ZSMPP 协议, 其安全性同样依赖于 SDH 假设. 然而, 上述协议功能复杂且执行过程中均需多次双线性配对运算, 因此不宜直接移植到资源受限的智能合约投票系统中.

本文主要目标是设计并实现基于零知识证明的智能合约投票系统. 通过构造轻量化 ZKDMP 协议来验证选票内容的有效性; 通过对选票进行同态加密操作从而保障选票内容的隐私. 该系统的以合约形式自动执行, 无需第三方机构参与. 本文主要贡献可总结如下:

(1)设计了智能合约投票系统, 该系统可对投票内容的有效性进行验证, 以避免无效选票对投票系统的影响. 投票过程无需第三方机构参与, 实

现去中心化投票以及自动化计票. 此外，通过 Java 开发工具将投票合约编译成 JAR 包文件，并将投票系统以合约代码的形式部署到区块链，通过智能合约实现整个投票过程自动化执行.

(2)为了在选票内容有效性验证过程中保护投票内容的隐私，本文构造了一个新的交互式零知识集合成员关系证明协议. 该协议通过两轮交互过程，实现证明者在不泄露投票内容具体信息的情况下，向验证者证明投票内容的有效性，即，投票内容是候选者集合中的一个. 此外，安全性分析表明，所提协议在离散对数困难假设下具有零知识性.

1 基于零知识证明的智能合约投票系统

1.1 系统框架

本文设计的智能合约投票系统是基于支持虚拟机的区块链环境下进行开发运行的，投票过程通过合约控制实现，并通过在智能合约中引入交互式零知识证明来保证投票数据验证过程的隐私性. 智能合约投票系统框架如图 1 所示，整个投票系统框架主要分为实现层、技术层和合约层三部分.

(1)实现层：实现层由区块链与基于 Java 的双线性配对密码库 (Java pairing based cryptography library, JPBC[24]) 两部分组成，其中区块链为智能合约的部署和执行提供了运行环境，JPBC 库则为基于配对的密码方案的实现提供了底层环境.

(2)技术层：技术层主要是提供了一种交互式零知识证明以及同态加密技术. 零知识证明技术用于投票内容的有效性验证，确保投票过程中投票内容的隐私性. 同态加密方案用于对选票加密，在计票阶段利用同态性直接对密文进行操作，避免解密运算，确保投票内容的隐私性.

(3)合约层：通过智能合约语言将投票系统以条款的形式进行描述，并通过 Java 开发工具将投票合约编译成计算机可执行的合约代码形式，最终实现智能合约投票系统按照预定规则自动化执行.

1.2 系统模型

本文设计的智能合约投票系统模型如图 2 所示，整个投票系统是基于支持虚拟机的区块链系统进行设计开发的，取代了以往的第三方投票机构，从而实现投票系统的去中心化.

智能合约投票系统执行流程如下：首先，投票发起者创建投票合约，并在合约中指定候选者名单，之后将合约部署到区块链中；投票者在完成注册后，可从区块链中调用投票合约来进行投票，并且在投票之前对选票内容的合法性进行验证，验证通过后将投票结果加密并按合约规定上传选

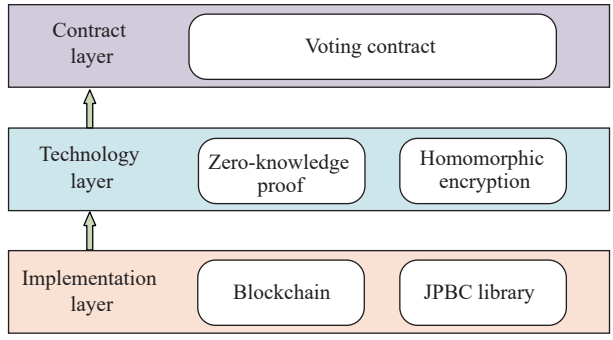

图 1 智能合约投票系统框架

Fig.1 Framework of the smart-contract voting system

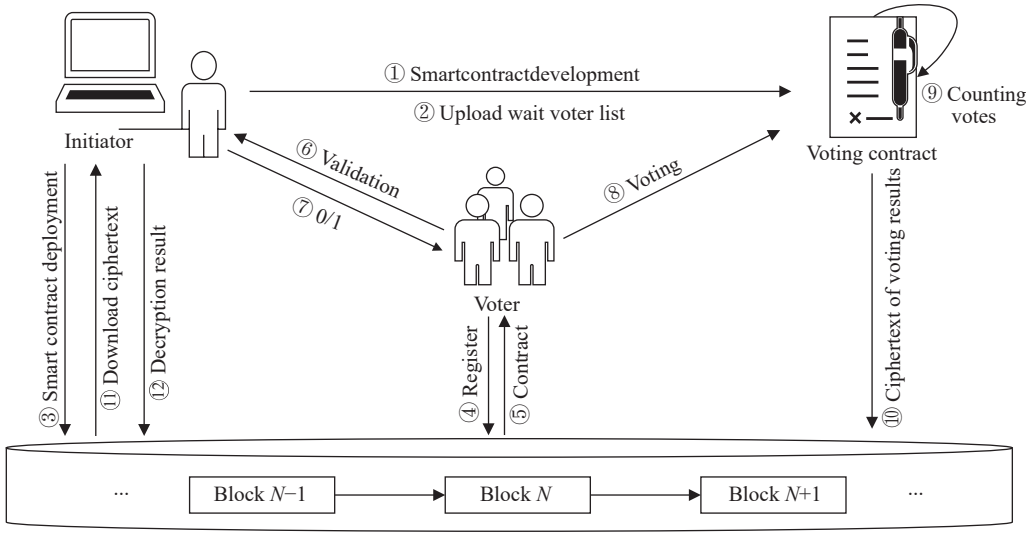

图 2 智能合约投票系统模型

Fig.2 Model of the smart-contract voting system

票; 当所有投票者完成投票后, 合约开始计票并将密文状态的计票结果上传至区块链; 计票结束后, 投票发起者将从链上获取密文状态下的计票结果, 利用自身私钥解密后, 将计票结果公布到区块链上.

投票系统是通过智能合约控制实现的, 投票合约在开始执行后, 整个投票流程会被自动触发执行, 智能合约投票的每一步都会被日志记录. 投票系统可分为初始化、注册、投票和计票四个阶段.

(1)初始化阶段: 系统初始化阶段首先借助 JPBC 库完成对系统参数的生成, 随后投票发起者制定投票智能合约并指定候选者名单, 随后将投票合约以 JAR 包的形式部署到区块链.

(2)注册阶段: 发起者在区块链上完成合约部署之后, 投票者需完成系统注册, 注册完成后, 投票者名单会被上传至区块链.

(3)投票阶段: 在投票之初, 发起者需对投票者的投票内容进行有效性验证, 只有验证通过, 投票才能继续进行. 最后, 投票者利用同态加密技术对验证通过的选票进行加密, 按合约规定以交易的形式上传密文状态的投票结果.

(4)计票阶段: 所有投票者完成投票后, 投票合约进行计票并将结果上传至区块链. 投票发起者可从区块链中获取密文状态的计票结果, 利用自身私钥进行解密后以交易的形式公布到区块链上.

2　零知识集合成员关系证明协议

零知识证明是指证明者在不向验证者透露秘密值的前提下, 使验证者相信自己确实持有该秘密. 基于此, 将零知识证明引入智能合约可保证合约中数据的隐私性, 同时由于智能合约具有触发自动执行的特点, 合约状态总是可以被保留下来, 这就为交互式零知识证明的实现提供了运行平台.

2.1　提出的协议

本文基于离散对数假设构造一个新的零知识集合成员关系证明协议 (ZSMPP). 该协议涉及证明者和验证者两方, 通过两方交互, 实现证明者在不透露任何消息的情况下, 向验证者证明其拥有的元素属于一个公开的集合, 具体协议描述如下.

设$\mathbf{Z}_p$是阶数为p的整数群, H为哈希算法, g是椭圆曲线上一个随机生成元, 公开集合 $\boldsymbol{M}$ 表示为 $\boldsymbol{M}=\{m_1,m_2,\cdots,m_n\}$. 证明者 P 欲向验证者 V 证明自己持有的元素m是公开集中的某一个, 即$m\in\boldsymbol{M}$, 但是不透露元素m本身的信息.

初始条件, 证明者 P 和验证者 V 共享消息集 $\boldsymbol{M}$, V 拥有安全哈希算 H, 交互过程如图 3 所示:

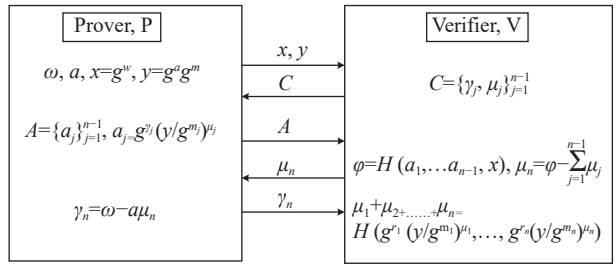

图 3　零知识集合成员关系证明协议

Fig.3　Zero-knowledge set membership proof protocol

(1) P→V: 证明者 P 首先选取两个随机数 $\omega,\alpha\in\mathbf{Z}_p$作为自己的私钥, 计算$Commit=(x,y)$, 其中 $x=g^{\omega}$, $y=g^{\alpha}g^{m}$, 然后将x和y发送给验证者;

(2) P←V: 验证者收到x和y之后, 随机选取 $\boldsymbol{C}=\{(\gamma_j,\mu_j)\}_{j=1}^{n-1}$作为 Challenge 发送给 P, 其中$\gamma_j$和$\mu_j$为$\mathbf{Z}_p$中的随机元素;

(3) P→V: 证明者 P 收到验证者发给的$\boldsymbol{C}$以后, 首先计算 $a_j=g^{\gamma_j}(y/g^{m_j})^{\mu_j}$, 其中 $m_j\in\boldsymbol{M}$, $m_j\neq m$且 $j=1,2,\cdots,n-1$, 然后将 $\boldsymbol{A}=\{a_j\}_{j=1}^{n-1}$作为 Response 发送给 V;

(4) P←V: 验证者 V 在收到 $\boldsymbol{A}$ 之后计算$\varphi=H(a_1,\cdots,a_{n-1},x)$以及$\mu_n=\varphi-\sum_{j=1}^{n-1}\mu_j$, 然后将$\mu_n$作为 Challenge 发送给 P;

(5) P→V: 证明者 P 收到以后计算$\gamma_n=\omega-\alpha\mu_n$, 随后将结算结果$\gamma_n$作为 Response 发送给验证者 V;

(6) V:验证者 V 收到γ_n后验证以下等式

$$\mu_1+\mu_2+\cdots+\mu_n=H(g^{\gamma_1}(y/g^{m_1})^{\mu_1},\cdots,g^{\gamma_1}(y/g^{m_n})^{\mu_n}) \tag{1}$$

如果验证结果失败, 验证者输出结果为“0”, 则表示验证者 V 将驳回 P 的证明并且结束方案; 如果检验成功, 结果返回为“1”, 则表示 V 将接收证明者的证明. 经过上述双方交互过程, P 向 V 证明了自己拥有秘密m但未向 V 透露关于m任何信息.

2.2　协议安全性分析

(1)完全性.

在提出的交互式证明协议中, 若证明者 P 的消息 m 确实属于消息集 $\boldsymbol{M}$ 且 P 能够遵守协议指令完成交互, 那么 V 总是能够接受 P 的证明.

证明: 设g是椭圆曲线群的一个生成元, 由协议中步骤(4)可知:

$$\mu_1+\mu_2\cdots+\mu_n=\varphi=H(a_1,\cdots a_{n-1},x) \tag{2}$$

在步骤(3)中计算时a_j规定$m_j \in M$且$m_j \neq m$，又由于P能够遵守协议指令完成交互，则必有$m = m_n$，又由于$y = g^{\alpha} g^{m}$以及$\gamma_n = \omega - \alpha\mu_n$，则有

$$g^{\gamma_n}(y/g^{m_n})^{\mu_n} = g^{\gamma_n}(g^{\alpha} g^{m_n}/g^{m_n})^{\mu_n} = g^{\gamma_n} g^{\alpha\mu_n} = g^{\gamma_n + \alpha\mu_n} = g^{\omega} = x \quad (3)$$

从而可以得出步骤(6)中验证等式成立.

(2)零知识性.

证明者P的私钥(ω,α)仅自己持有，在交互式证明过程中，证明者将包含消息的承诺$y = g^{\alpha} g^{m}$发送给验证者V，由于计算离散对数问题是困难的，因此在交互过程中，除了关于消息m的承诺之外，证明者无法获取其他任何信息，这保证了整个交互过程中消息m的零知识性.

3 投票智能合约

在设计和实现智能合约投票系统之前，我们首先给出具体投票合约. 本节将采用智能合约规范语言SPESC[25]，对投票系统的初始化、注册、投票和计票四个阶段的触发条件以及执行过程进行合约化描述.

如图4所示，投票智能合约包括发起者(Initiator)和投票者(Voter)两个参与方. 该合约包含5个条款(Term)，条款1表示在投票发起人进行参数初始化(initParam)之后，可指定候选者名单(对应candidateForm算法)；条款2表示在发起者指定候选者名单之后，投票者进行注册(voterRegist)；条款3描述的是在投票者注册完成之后，必须与发起者进行交互式ZSMPP来验证选票的有效性；条款4表示当ZSMPP输出为“1”时投票者利用Boneh、Goh和Nissim提出的加密算法[26](简称BGN算法)对投票内容进行加密；条款5表示发起人在所有投票者完成投票之后，合约进行计票(对应voteResult算法)，最终发起者对计票结果进行解密获取投票最终结果. 合约中条款仅描述了投票各个阶段的触发条件以及执行的算法，具体算法描述将在下节中给出.

智能合约最终将被部署到区块链平台进行自动化执行，如图5所示，部署过程可分成两个阶段：首先利用Java开发工具将投票系统中的相关算法编译成Java Archive (JAR)格式的智能合约代码；随后，将JAR包上传至区块链网络. 在发起者和投票者的共同参与下，最终合约按照预定规则自动执行.

4 智能合约投票系统的设计与实现

本节将具体介绍智能合约投票系统以及实现过程，为了便于后续叙述，首先给出投票系统中涉及的部分符号及其相关说明，具体如表1所示.

```
1.  contract voting{
2.    party initiator{
3.      candidateForm()
4.      initParam()
5.      Decyrpt()
6.    }
7.    party voter{
8.      voterRegist()
9.      BNG()
10.   }
11.   term term1 : initiator can candidateForm,
12.       when after initiator did initParam.
13.   term term2 : voter can voterRegist,
14.       when after initiator did candidateForm.
15.   term term3 : initiator and voter must do ZSMPP,
16.       when after voter did voterRegist.
17.   term term4 : voter can BNG,
18.       while ZSMPP output 1.
19.   term term5 : initiator can voteResult,
20.       when after all voters did BNG,
21.       while decrypt voteResult.
22. }
```

图4 SPESC语言编写的投票智能合约

Fig.4 Voting contracts written in SPESC language

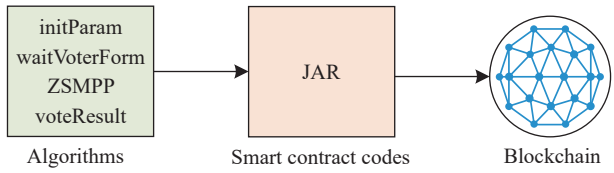

图5 智能合约部署流程

Fig.5 Deployment process of smart contract

表1 符号说明表

Table 1 Notation declaration

Symbol	Description
$\boldsymbol{G}$	Elliptic curve multiplicative cyclic group
$\boldsymbol{Z}$	Integer group
g	Generator of the group $\boldsymbol{G}$
n	Number of candidates
s	Number of voters
v_i	The i-th voter
sk_i	Secret key of voter v_i
pk_i	Public key of voter v_i
num_i	Voting number of voter v_i
w_j	The j-th candidate
wid_j	Identity of candidate w_j

本节将基于区块链系统环境，根据投票系统所需的初始化、注册、投票以及计票四个部分，对智能合约投票系统的设计以及实现流程进行详细的阐述.

4.1　初始化阶段

初始化阶段由合约发起者执行, 如图 6 所示. 首先需要完成系统参数生成, 生成椭圆曲线乘法循环群$\boldsymbol{G}$和整数群$\mathbf{Z}$, 再根据群$\boldsymbol{G}$产生生成元$g \in \boldsymbol{G}$. 群$\boldsymbol{G}$和$\mathbf{Z}$的生成是基于 JPBC 库实现的, 其中 JPBC 中的椭圆曲线选取 Type A 类型, 如表 2 中算法所示.

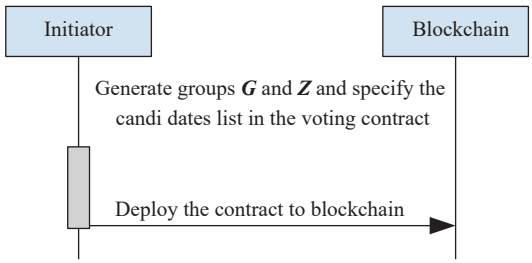

图 6　智能合约投票系统初始化阶段

Fig.6　Initialization of the smart-contract voting system

表 2　initParam 算法

Table 2　initParam algorithm

input: r_{bit}, q_{bit}

output: void

pg ← new TypeACurveGenerator(r_{bit},q_{bit})

typeA**Params** ← pg.generate()

pairing←PairingFactory.getPairing(typeA**Params**)

$\boldsymbol{G}$ ← pairing.get$\boldsymbol{G}$1();

$\mathbf{Z}$ ← pairing.get$\mathbf{Z}$r();

g ← $\boldsymbol{G}$.newRandomElement().getImmutable();

return;

该算法以群$\boldsymbol{G}$的阶数 r_{bit} 和$\mathbf{Z}$的阶数 q_{bit} 作为输入, 通过 TypeACurveGenerator() 生成 TypeA 生成器, 利用 TypeA 生成器得到双线性对参数, 再通过 getPairing() 获取双线性对, 最后由双线性对生成椭圆曲线乘法循环群和整数群, 生成元g是在椭圆曲线群上随机生成的.

接下来, 合约发起者需在合约中指定候选者名单, 如表 3 所示. 该算法输入一个数组 **params**, 数组中的每个元素对应候选者w_j的地址, 该地址是候选者的唯一标识. 假设有 n 个候选者, 则算法 newNum 返回一个整数 z 且满足$2^z > n$, 第 k 位候选者的投票编号为$2^{k\cdot z}$, 例如, 当投票者选择候选者 1 时, 对应的候选者的编号是2^z, 算法返回候选者的候选列表 candidateList, 大小为 n, 候选列表中包括候选者的地址, 编号以及所得的票数(初始化值为 0), 并且会在结果集 resultMap 中存放所有候选者及对应的票数.

表 3　candidateForm 算法

Table 3　candidateForm algorithm

input: **params**

output: candidateList

params={}

for i ← 0 **to** params.length **by** 1 **do**

n ← params.length;

w.setCandidateAddress(params[i]);

z← newNum();

w.setNumid((int)pow(2,(i+1)·z);

w.setNumid(i+1);

w.setTotal(0);

candidateList.add(w);

resultMap.put(params[i−1],0);

end for

return candidateList;

候选者名单指定完成后, 合约发起者需将投票合约以 JAR 包的形式部署到区块链上. 图 7 为智能合约发布的结果, 其中“address”表示合约地址字段, “txid”表示合约交易 id 字段.

```
zzy@ubuntu :~/code/githubcode/JoTochain/src$ ./multichain-cli chain0001 publishcontractfrom1KLKpKRgbGcR3aJ7ankjr6dxXAfbwLJddGGhD contract /home/zzy/javaContractModel.jar
{"method" : "publishcontractfrom", "params": ["1KLKpKRgbGcR3au7ankjr6dxXAfbwl.ddGctThD", "contract", "/home/zzy/javacontractModel.jar"], "id" : 1, "chain_name" : "chain0001"}
{
        "address" : "1322ei97hJqioxSDxspv5U1uZP3ZrXmY6HeDq6",
        "txid": "106b370a6144c8fd5eb07caa3c49fb38b84a967b8fde74198c47ec083"
}
```

图 7　智能合约发布结果

Fig.7　Results of smart contract release

图 8 描述了初始化合约结果, 合约部署完成且初始化后, 可看到当前投票者和候选者的信息. 字段“votersData”代表投票者数据, 其中字段“voterTxid”为投票者交易的 id, 从交易中可获取投票者名单的信息, 在合约初始后, 投票者名单为空. 字段“candidateData”代表候选者信息, “candidateTxid”代表候选者交易 id, 查看交易 id 可获取候选者名单信息.

```
{"method" : "executecontract", "params" : ["1THQ:EVKpgQUHARXedvViAA4hfL5V921MN7t9DRpj" ," _init"], "id":1, "chain_name" : "chain0001
    "txid" : "5219934338e9f47dd1cbaba087e9968e52830aeab6dba74aa8deef0",
    "votersData" : {
        "voterTxid": 8de099b788d4140763a232a043c39eecfc6202d687e0ee386"
    },
    "candidateData" : {
        "candidateTxid": 2a015d17a70632e7ca67b9367c90c60b53bf148be72c"
    }
}
```

图 8　智能合约初始化结果

Fig.8　Results of smart contract initialization

4.2 注册阶段

当投票合约在区块链上完成部署后，投票者v_i若想获得投票资格，则需进行注册．如图 9 所示，投票者需先进行注册并获取合约，投票者可从区块链上查询当前人员信息，包括获取到当前投票者和候选者名单．

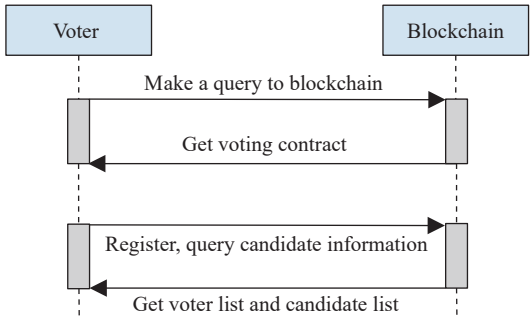

图 9 智能合约投票系统注册阶段

Fig.9 Registration of the smart-contract voting system

投票者的注册过程如表 4 中算法所示，算法输入为数组 params，数组中存放着作为投票者的唯一标识地址，最后算法返回投票者列表 voterList．注册完成之后，列表中将包含投票者地址以及投票者的投票状态（初始值为 false），此时投票者可从区块链中查询候选人名单和当前投票者名单．

表 4 voterRegist 算法

Table 4 voterRegist algorithm

input: **params**

output: voterList
params={}
for *i* ← 0 **to** params.length **by** 1 **do**
    v ← new Voter();
    v.setVoterAddress(params[*i*]);
    v.setVoteStatus(false);
    voterList.add(*v*);
end for
return voterList;

4.3 投票阶段

投票者从区块链上获取候选人信息后，开始进行投票．投票过程借助于区块链完成，如图 10 所示，在投票的前期，投票者v_i生成公私钥对$< sk_i, pk_i >$，公钥是椭圆曲线乘法循环群$\boldsymbol{G}$中的元素，私钥是整数群 $\mathbf{Z}$ 中的元素．在投票过程中，需对投票者的投票内容进行有效性验证，验证过程调用交互式零知识集合成员关系证明协议 ZSMPP，协议经过两轮交互对投票内容有效性进行验证．在交互过程中，投票者相当于协议中的证明者，合约发起者相当于协议中的验证者．接下来对投票阶段的参数生成、承诺生成、两轮交互以及验证给出具体介绍．

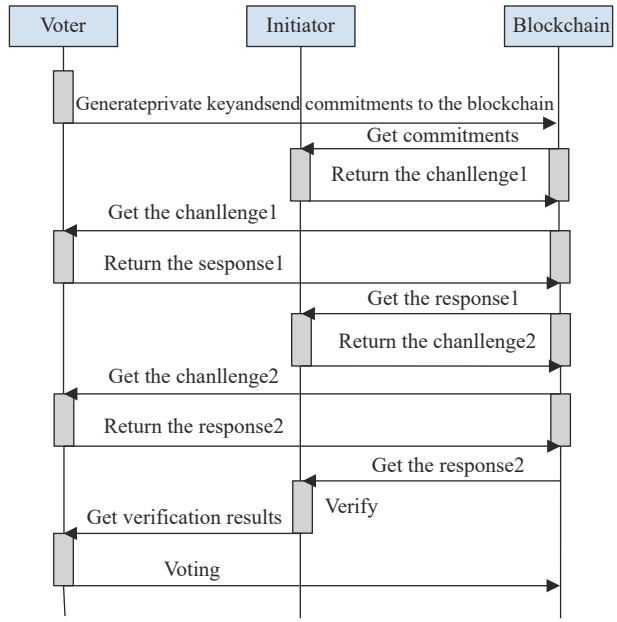

图 10 智能合约投票系统投票阶段

Fig.10 Voting of the smart-contract voting system

令$\boldsymbol{M} = \{m_1, m_2, \cdots, m_n\}$投票内容 (候选者) 组成的集合，其中$m_i$是候选者的编号，接下来投票者$v_i$证明自己的投票内容$num_i$属于投票集合$\boldsymbol{M}$．投票者首先利用自己的公私钥生成承诺，如表 5 中算法所示，输入投票者的地址 address 以及投票者的投票号 num，其中地址作为投票者的身份标识，根据地址判断投票者是否在注册者名单内，若不在名单内，则算法终止；若在注册名单中，则可以开始投票．投票者生成自己的私钥$\mathrm{sk} = (\mathrm{sk}_1, \mathrm{sk}_2)$以及公钥$\mathrm{pk} = (\mathrm{pk}_1, \mathrm{pk}_2)$，其中$\mathrm{pk}_1 = g^{\mathrm{sk}_1}$，$\mathrm{pk}_2 = g^{\mathrm{sk}_2}$，根据投票者的投票号对应候选者编号，newNum 函数返回一个整数z且满足$2^z > n$，例如，当投票者的投票号为 2 时，对应的候选者的编号 wid 为$2^{2\cdot z}$．最后，该算法生成承诺$Commit = (x, y) = (\mathrm{pk}_1, \mathrm{pk}_2 \cdot g^{\mathrm{wid}})$并将其以交易的形式存放到区块链中．

合约发起者在查询区块链得到承诺后，开始执行 ZSMPP 中的第一轮交互并生成对应挑战码$\boldsymbol{C} = \{(\gamma_j, \mu_j)\}_{j=1}^{n-1}$，其中$\gamma_j, \mu_j \in \mathbf{Z}_p$．如图 11 所示，在智能合约输入中，generateChallenge1 为函数，输入参数为合约发起者的地址，在合约输出字段中，“userResult”表示第一轮挑战码，其中$\boldsymbol{R} = \{\gamma_1, \gamma_2, \cdots \gamma_{n-1}\}$，$\boldsymbol{U} = \{\mu_1, \mu_2, \cdots \mu_{n-1}\}$．挑战码生成后将发布到区块链中．

表 5　generateCommit 算法

Table 5　generateCommit algorithm

input: address,num

output: Commit

Commit ← {};

if address **in** voterList **then**

　　sk1←Z.newRandomElement().getImmutable().duplicate();

　　sk2←Z.newRandomElement().getImmutable().duplicate();

　　pk1 ← g.duplicate().powZn(sk1);

　　pk2 ← g.duplicate().powZn(sk2);

　　x ← pk1.duplicate();

　　yy ← pk2.duplicate();

　　wid ← (int) pow(2, num * newNum());

　　e ←g.duplicate().pow(BigInteger.valueOf(wid));

　　y ← yy.mul(e);

　　Commit.add(x);

　　Commit.add(y);

end if

return Commit;


```
"contractInput" : {
    "address" : "1THQwEVKpQUHAXedVKAA4hfFL5V92LM7t9DRpj",
    "latestCodeId":eef249a6ddf708ef4e39aa6ad3384e9a2b742ae26d62ac79e71",
    "latestExecuteId":32cac35bc8355ccb451b50797536d8e01cc01bfe466653116",
    "userParams" : "[\"1KLKpKRgbGcR3aJ7ankjr6dxXAfbwLJddGGJhD\"]",
    "fromAddress" : "1KLKpKRgbGcR3aJ7ankjr6dxXAfbwLJddGGJhD",
    "funName" : "generatechallenge1"
},
"contractOutput" : {
    "userResult":  "R=:[[36,20,50,60,-10,16,110,47,13,89,-35,96,45,109,78,84,-108,
-65,-127,36],[37,-117,63,-41,63,30,-125,112,118,-2,109,-95,-31,20,108,8,30,-97,-
33,18], [52,44,-14,-22,-84,4,10,55,-83,14,82,-26,123,-9,-112,104,87,-43,45, 351], U=:
[[96,95,-20,64,89,-37,-19,-64,-38,52,-105,-106,-52,37,5,-91,-123,31,
93,48],[69,  60,115,43,49,112,80,126,-116,91,112,-20,19,127,9,-92,-21,-100,-
116,4],[66,-46,75,100,60,92,94,-105,96,-39,57,119,35,127,119,-115,-48,-53,-18]"
}
```


图 11　第一轮挑战码对应的交易

Fig.11　Transaction of the first challenge

投票者从区块链中获取到第一轮挑战码后，生成第一轮响应码$\boldsymbol{A}=\{a_j\}_{j=1}^{n-1}$，随后执行 ZSMPP 中的第二轮交互，合约发起者计算$\varphi=H(a_1,\cdots,a_{n-1},x)$以及$\mu_n=\varphi-\sum_{j=1}^{n-1}\mu_j$，其中$H$是哈希函数，生成第二轮挑战码 miuN 即$\mu_n$.

如表 6 中算法所示，算法 generateChallenge2 计算集合$\boldsymbol{A}$中所有元素与x的和，利用哈希函数对和求哈希，最后返回第二轮的挑战码 miuN，然后发布交易到区块链. 投票者从区块链中获取第二轮的挑战码之后，利用自身私钥$\mathrm{sk}=(\mathrm{sk}_1,\mathrm{sk}_2)$计算$\gamma_n=\mathrm{sk}_1-\mathrm{sk}_2\mu_n$并生成第二轮挑战码 gamaN($\gamma_n$). 如图 12 所示，在合约输入中，generateResponse2 为函数，输入参数是投票者的地址，合约将输出第二轮的响应码 gamaN. 最后，合约发起者收到第二轮响应码之后，会验证等式

$$\mu_1+\mu_2+\cdots+\mu_n=H(g^{\gamma_1}(y/g^{m_1})^{\mu_1},\cdots,g^{\gamma_n}(y/g^{m_n})^{\mu_n}) \tag{4}$$

表 6　generateChallenge2 算法

Table 6　generateChallenge2 algorithm

input: address

output: miuN

for i ← 0 **to** n-1 **by** 1 **do**

　　sum ← sum.add(A.get(i));

end for

sum ← sum.add(x));

phi ← hash(sum);

for i ← 0 **to** n-1 **by** 1 **do**

　　miuN ← miuN.sub(U.get(i));

end for

return miuN;


```
"contractInput" : {
    "address" : "1THQwEVKpQUHAXedvKAA4hfFL5V92LM7t9DRpj",
    "latestcodeId": "eef249a6ddf708ef964e39aa384a2b7d05ae26d62ac9b79e71",
    "latestExecuteId": "14b2541b6582758ade4fd7bff5738c5362e56be493d7bed",
    "userParams" : "[\"1KLKpKRgbGcR3aJ7ankjr6dxXAfbwLJddGGJhD\"]",
    "fromAddress" : "1KLKpKRgbGcR3aJ7ankjr6dxXAfbwLJddGGJhD",
    "funName": "generateResponse2"
},
"contractOutput" : {
   "userResult":"gamaN=:[51,44,-16,-60,-37,78,77,-55,-67,88,-22,103,-63,-64,26,-59,
   117,-42,-3,-31]"
}
```


图 12　第二轮响应码对应的交易

Fig.12　Transaction of the second response

对投票者投票内容的有效性进行验证. 若验证失败，则终止投票；否则，投票者对投票内容进行加密并将投票状态改为 true.

4.4　计票阶段

当选票有效性通过验证后，投票者将通过 BGN 同态加密算法[26] 对其进行加密处理. 如图 13 所示，BGN(w_j)表示第 i 位投票者对第 j 位候选者投票结果对应的密文. 当所有投票者完成投票后，合约将执行计票程序.

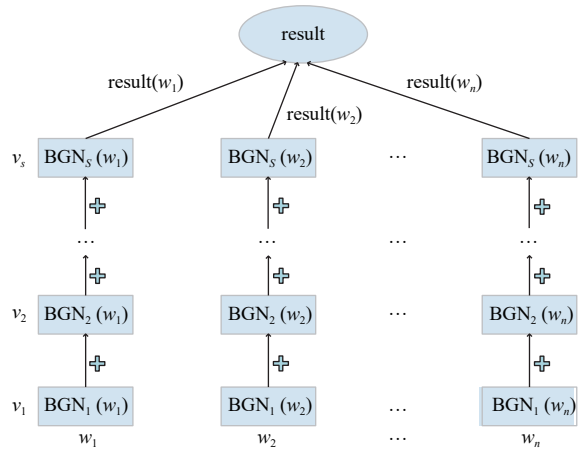

图 13　智能合约投票系统计票阶段

Fig.13　Vote-counting of the smart-contract voting system

表 7 中算法描述了投票者对投票内容加密以及合约计票过程. 其中 BGN 算法满足加法同态性, 即$\mathrm{BGN}(w_1)+\mathrm{BGN}(w_2)=\mathrm{BGN}(w_1+w_2)$. 合约统计所有候选者$w_j$的最终投票结果表示为$\mathrm{result}(w_j)=\mathrm{BGN}_1(w_j)+\mathrm{BGN}_2(w_j)+\cdots+\mathrm{BGN}_s(w_j)$并上传到区块链上.

表 7 voteResult 算法

Table 7 voteResult algorithm

```
input: candidateList
output: result
for j ← 0 to n-1 by 1 do
    candidateddress ← candidateList.get(j).candidateaddress;
    for i ← 0 to s-1 by 1 do
      eTotal ← eTotal +
      BGN(getPK.initiator, candidateaddress);
    end for
    result.put(candidateaddess, eTotal);
end for
return result;
```

5 投票协议安全特性与性能分析

5.1 安全特性分析

接下来将对本文提出的基于零知识证明的智能合约投票系统安全特性进行分析, 具体如下:

(1)选票有效性.

指选票内容是候选者集合中的一员, 通过执行零知识集合成员证明协议, 判定选票内容有效性, 若选票内容不属于候选者集合, 则终止投票活动.

(2)选票隐私性.

主要考虑所选候选人的隐私. 在有效性验证阶段, ZSMPP 协议的零知识性可以保证验证过程不泄露选票信息. 此外, 计票是对密文状态下的选票进行同态运算, 选票的隐私性基于安全的 BGN 同态加密算法实现.

(3)唯一性.

每个注册投票者初始投票状态都是 false, 在完成投票之后该状态变成 true, 整个过程都以交易的形式被记录在区块链中, 由区块链的不可篡改性保证了每个投票者只能参与一次投票.

(4)无监督性.

投票协议以智能合约的形式被部署到区块链平台并按照预定规则自动执行, 由于区块链的不可篡改性和共识机制保证预定规则不变性和规则执行的正确性, 从而确保投票合约中仅需发起者和投票者两个实体, 无需可信监票人参与.

(5)自计票性.

协议计票过程由智能合约按照事先制定的规则自动化执行, 当触发计票条件, 即所有投票者完成投票后, 合约自动执行 voteResult 算法对密文状态的选票进行同态运算并将密态计票结果上传至区块链.

表 8 展示了本文提出的投票协议与现有 4 个投票方案之间关于安全特性的对比, 从结果中不难发现, 仅有文献 [10] 和本文提出的方案能够实现对投票内容的有效性验证, 然而该文投票过程需要在可信监票人的参与下完成. 文献 [11] 提出的投票方案虽然能够追踪到实施重复投票的攻击者, 但牺牲了投票的唯一性特征. 此外, 本文计票过程由智能合约程序自动完成, 而文献 [10 ~ 12] 则需要投票发起者或监票人参与才能实现.

表 8 不同方案之间安全特性对比

Table 8 Comparison of security features

Voting schemes	Validity of the ballot	Privacy of ballots	Uniqueness	Supervision free	Self-counting
[10]	Yes	Yes	Yes	No	No
[11]	No	Yes	No	Yes	No
[12]	No	Yes	Yes	Yes	No
[27]	No	Yes	Yes	No	No
Ours	Yes	Yes	Yes	Yes	Yes

5.2 性能分析

本节将通过仿真实验对提出的智能合约投票协议的性能进行分析, 实验运行在 2.6 GHz 英特尔 CPU 和 8 GB 内存的 64 位 Ubuntu 16.04 操作系统上. 基于交互式零知识证明的的智能合约投票系统运算效率主要取决于投票阶段和计票阶段, 本节将从以下两方面来对投票系统进行性能评估, 一是针对不同的投票者数量, 对投票系统的投

票和计票阶段进行耗时对比; 二是针对不同的候选人数量, 对投票系统的投票和计票阶段进行耗时对比.

图 14 展示的是当候选者数量为 5 时, 对不同数量投票者在投票阶段和计票阶段的耗时对比图, 从图中可以发现, 当投票者数量达到 500 时, 投票和计票过程均能在 8000 ~ 9000 ms 之间完成. 图 15 展示的是当投票者数量为 100 时, 针对不同数量的候选者, 投票阶段和计票阶段的耗时对比图, 当候选者数目为 20 时, 投票和计票时间分别是 7784 ms 和 9218 ms. 实验结果表明, 随着投票者或候选者数量的增加, 投票与计票阶段的耗时均呈线性增长, 智能合约投票系统能够高效实施.

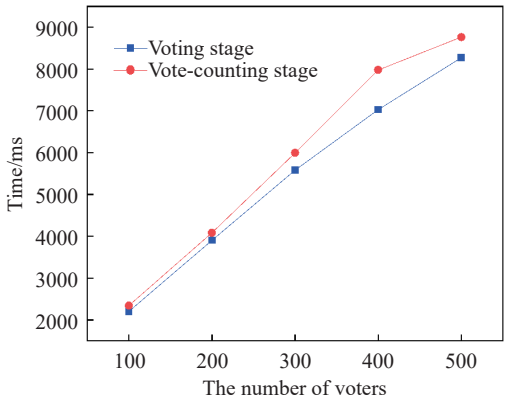

图 14　不同数量投票者耗时对比

Fig.14　Time cost of different numbers of voters

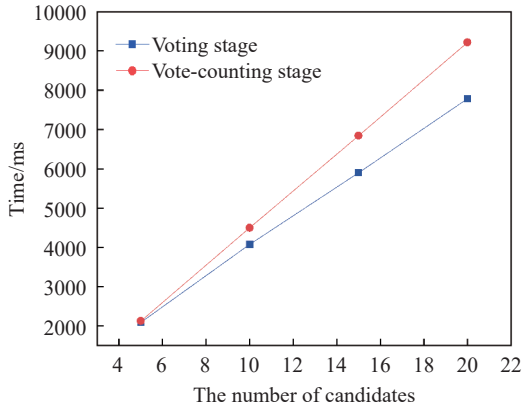

图 15　不同数量候选者耗时对比

Fig.15　Time cost of different numbers of candidates

6　结论

本文设计了基于交互式零知识证明的智能合约投票系统, 实现投票过程的去中心化以及计票过程的自动化. 基于离散对数困难假设, 本文构造了一个新的交互式零知识集合成员关系证明协议, 通过将该协议引入到投票合约的设计中, 实现投票者在不透露投票内容的前提下向发起者证明投票内容的有效性. 此外, 通过将投票合约编译成 JAR 包, 实现智能合约投票系统在区块链中的部署和自动化执行, 保证了投票数据的有效性和隐私性. 实验结果表明, 本文提出的智能合约投票系统在投票和计票阶段均可高效实施.

参　考　文　献

[1] Wang D, Zhu Y, Chen E, et al. Smart legal contract and its research progress. *Chin J Eng*, 2022, 44(1): 68
(王迪, 朱岩, 陈娥, 等. 智能法律合约及其研究进展. 工程科学学报, 2022, 44(1): 68)

[2] Wang S, Ouyang L, Yuan Y, et al. Blockchain-enabled smart contracts: Architecture, applications, and future trends. *IEEE Trans Syst Man Cybern Syst*, 2019, 49(11): 2266

[3] Zhu Y, Wang Q S, Qin B H, et al. Survey of blockchain technology and its advances. *Chin J Eng*, 2019, 41(11): 1361
(朱岩, 王巧石, 秦博涵, 等. 区块链技术及其研究进展. 工程科学学报, 2019, 41(11): 1361)

[4] Zhu Y, Qin B H, Chen E, et al. An advanced smart contract conversion and its design and implementation for auction contract. *Chin J Comput*, 2021, 44(3): 652
(朱岩, 秦博涵, 陈娥, 等. 一种高级智能合约转化方法及竞买合约设计与实现. 计算机学报, 2021, 44(3): 652)

[5] Hewa T, Ylianttila M, Liyanage M. Survey on blockchain based smart contracts: Applications, opportunities and challenges. *J Netw Comput Appl*, 2021, 177: 102857

[6] Buterin V. A next-generation smart contract and decentralized application platform [R/OL]. *Ethereum* (2014-12-01) [2022-07-07].https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf

[7] Lerner S D. RSK Bitcoin powered smart contracts [R/OL]. *Sciencepaper Online* (2015-11-19) [2022-07-07]. http://cryptochainuni.com/wp-content/uploads/Rootstock-WhitePaper-v9-Overview.pdf

[8] Androulaki E, Barger A, Bortnikov V, et al. Hyperledger fabric: A distributed operating system for permissioned blockchains // *Proceedings of the Thirteenth EuroSys Conference. Portugal*, 2018: 1

[9] Chaum D L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun ACM*, 1981, 24(2): 84

[10] Zhang P, Yu J P, Liu H W. A homomorphic signcryption scheme and its application in electronic voting. *J Shenzhen Univ Sci Eng*, 2011, 28(6): 489
(张鹏, 喻建平, 刘宏伟. 同态签密方案及其在电子投票中的应用. 深圳大学学报(理工版), 2011, 28(6): 489)

[11] Sun M, Wang Y B. Traceable anonymous electronic voting scheme based on blockchain. *Cyberspace Secur*, 2019, 10(9): 85
(孙萌, 王昀飚. 基于区块链的可追踪匿名电子投票方案. 网络空间安全, 2019, 10(9): 85)

[12] Zheng J, Lai H C. Blockchain e-voting scheme based on one-time

ring signature. *Appl Res Comput*, 2020, 37(11): 3378
(郑剑, 赖恒财. 基于一次性环签名的区块链电子投票方案. 计算机应用研究, 2020, 37(11): 3378)

[13] Satizábal C, Páez R, Forné J. Secure Internet Voting Protocol (SIVP): A secure option for electoral processes. *J King Saud Univ Comput Inf Sci*, 2022, 34(6): 3647

[14] Wang K H, Mondal S K, Chan K, et al. A review of contemporary e-voting: Requirements, technology, systems and usability. *Data Sci Pattern Recognit*, 2017, 1(1): 31

[15] Alvarez R M, Hall T E, Trechsel A H. Internet voting in comparative perspective: The case of Estonia. *PS Political Sci Politics*, 2009, 42(3): 497

[16] Zhao Z C, Chan T H H. How to vote privately using bitcoin // *International Conference on Information and Communications Security*. *Beijing*, 2015: 82

[17] Tarasov P, Tewari H. Internet voting using zcash [R/OL]. *Sciencepaper Online* (2017-06-20) [2022-07-07].https://eprint.iacr.org/2017/585

[18] McCorry P, Shahandashti S F, Hao F. A smart contract for boardroom voting with maximum voter privacy // *International Conference on Financial Cryptography and Data Security*. *Sliema*, 2017: 357

[19] Yu B, Liu J K, Sakzad A, et al. Platform-independent secure blockchain-based voting system // *International Conference on Information Security*. *Guildford*, 2018: 369

[20] Camenisch J, Chaabouni R, Shelat A. Efficient protocols for set membership and range proofs // *International Conference on the Theory and Application of Cryptology and Information Security*. *Melbourne*, 2008: 234

[21] Morais E, Koens T, Van W C, et al. A survey on zero knowledge range proofs and applications. *SN Appl Sci*, 2019, 1(8): 946

[22] Boneh D, Boyen X. Short signatures without random oracles // *International Conference on the Theory and Applications of Cryptographic Techniques*. *Interlaken*, 2004: 56

[23] Yin H J, Chen E, Zhu Y, et al. An efficient zero-knowledge dual membership proof supporting pos-and-neg membership decision. *Mathematics*, 2022, 10(17): 3217

[24] De Caro A, Iovino V. jPBC: Java pairing based cryptography // 2011 *IEEE Symposium on Computers and Communications*. *Corfu*, 2011: 850

[25] He X, Qin B H, Zhu Y, et al. SPESC: A specification language for smart contracts // 2018 *IEEE* 42*nd Annual Computer Software and Applications Conference*. *Tokyo*, 2018, 1: 132

[26] Boneh D, Goh E J, Nissim K. Evaluating 2-DNF formulas on ciphertexts // *Proceedings of the* 2005 *Second International Conference on Theory of Cryptography*. *Spain*, 2005: 325

[27] Kumar M, Chand S, Katti C P. A secure end-to-end verifiable internet-voting system using identity-based blind signature. *IEEE Syst J*, 2020, 14(2): 2032