边缘计算模式下全同态加密智能合约设计

杨亚涛[1,2], 林天祥[1], 陈剑源[1], 曾　萍[1,2], 刘培鹤[1]

[1]北京电子科技学院电子与通信工程系 北京 中国 100070
[2]西安电子科技大学通信工程学院 西安 中国 710071

摘要　区块链产生的成千上万的交易信息数据不断被收集到区块链中, 会给区块链网络的计算资源和存储空间造成极大负担。为了解决区块链网络计算资源不足和存储空间有限的问题, 我们将边缘计算模式应用于联盟链 Hyperledger Fabric 系统中, 在区块链中设计了基于边缘计算模式的智能合约, 以提高区块链的存储空间和运行效率。在此基础上, 针对区块链存在的数据和用户身份隐私的风险, 提出了基于边缘计算模式的全同态加密智能合约。在执行智能合约过程中, 调用了微软的全同态加密 SEAL 库, 边缘节点的身份信息和区块链的交易数据受到全同态加密算法 BFV 的保护, 在保持交易数据可计算性的同时, 实现了交易信息和边缘节点身份的隐私保护。最后, 经过测试和分析, 边缘计算模式下的原型系统在 14 个边缘节点同时访问时的单节点平均访问时间仅为 139.68 ms, 相比于非边缘计算模式下的原型系统具有更好的性能; 相比目前响应较快的加密联盟链系统 FabZK, 本系统平均用时减少 35.84%, 相比效率提升 55.86%。设计的边缘计算模式下的全同态加密智能合约可以有效提高 Hyperledger Fabric 系统中网络的运行效率, 同时减少了对存储空间的要求, 对于实际环境下的区块链部署, 具有一定参考价值。

关键词　边缘计算; 智能合约; 全同态加密; 隐私保护; 区块链
中图法分类号　TP309　DOI 号 10.19363/J.cnki.cn10-1380/tn.2022.03.10

Smart Contract with Fully Homomorphic Encryption under Edge Computing Mode

YANG Yatao[1,2], LIN Tianxiang[1], CHEN Jianyuan[1], ZENG Ping[1,2], LIU Peihe[1]

[1]Department of Electronic and Communication Engineering, Beijing Electronics Science and Technology Institute, Beijing 100070, China
[2]School of Telecommunication Engineering, Xidian University, Xi'an 710071, China

Abstract　Thousands of transaction data generated in the blockchain are continuously collected into the blockchain, the computing resources and storage space in blockchain network are becoming great burden. In order to solve the problem of insufficient computing resources and limited storage space in blockchain network, the edge computing model has been introduced to Hyperledger Fabric consortium blockchain system, a smart contract based on edge computing is designed to improve the storage space and operation efficiency in blockchain. After that, in view of the risk of data and user identity privacy in blockchain, a fully homomorphic encryption smart contract based on edge computing mode is proposed. In the process of executing the smart contract, Microsoft's fully homomorphic encryption seal library is adopted. The identity information of edge nodes and transaction data in blockchain are protected by BFV fully homomorphic encryption algorithm. It can not only maintain the transaction data's computability, but the privacy preservation of transaction data and edge nodes' identity information also can be guaranteed. Finally, after being tested and analyzed, the average communication time of a single node in the prototype system under the edge computing mode is only 139.68 ms when 14 edge nodes are accessing at the same time, which is much better than the prototype system under the non edge computing mode. Compared with the Fabzk consortium blockchain system with excellent computing performance, the average access time in our prototype system is reduced by 35.84%, and the efficiency is improved by 55.86%. This smart contract with fully homomorphic encryption under edge computing mode can increase the working efficiency of Hyperledger Fabric consortium blockchain system, furthermore, the requirement for storage space will be lower, which provides valuable preferences for blockchain system deployment in real network scene.

Key words　edge computing; smart contract; full homomorphic encryption; privacy preservation; blockchain

通讯作者: 杨亚涛, 男, 1978 年生, 博士, 教授, 主要研究方向为密码学与通信安全, Email: yy2008@163.com。
本课题得到北京高校高精尖学科建设项目(No.3201023), "十三五"国家密码发展基金(No.MMJJ20170110)资助。
收稿日期: 2021-01-10; 修改日期: 2021-03-25; 定稿日期: 2022-01-10

1 引言

联盟链是介于公有链和私有链之间的区块链, 对链上的所有成员均有约束和限制, 具有较好的安全性, 主要适用于多个机构之间的协作[1]。随着过多用户的访问, 联盟链网络也会产生很大的通信负担和延迟, 为此, 将边缘计算模式[2]引入联盟链可以较好缓解网络负载问题。Xiong 等人[3], 提出了一个在资源受限的设备中使用区块链边缘计算的原型, 他们展示了移动设备在边缘计算模式下利用计算资源来促进区块链应用的场景。Liu 等人[4]提出了一个使用边缘计算的资源分配框架, 以便在视频流系统中使用区块链进行转码, 通过设计激励机制, 他们考虑了两种方法来减轻边缘计算环境下的转码任务负担。文献[5]提出了边缘计算环境下跟踪车辆节点数据的区块链协同认证方案, 该方案减少了中心节点的计算负担, 具有较高的安全性。

联盟链中的所有节点均可访问联盟链中的交易数据和内容, 不可避免地对联盟链中各组织的交易数据造成较大的安全风险。通常在联盟链上账本中的数据具有较强的数学特性, 采用一般加密方式会破坏联盟链链上数据的计算性。而采用全同态加密技术[6]可以有效地保障数据的隐私性和数学特性。Xiao 等人[7]提出了一种基于同态加密的可信区块链交易认证方案, 结合保险业务分布式权限管理的特点, 采用数据一致性、智能合约和特殊的链式存储结构, 实现了链上数据的共享。Marella 等人[8]提出了一种分布式电子投票系统 GenVote, 在以太坊区块链的平台上利用同态加密来实现透明投票过程, 同时保护了计票和选民的身份隐私。文献[9]利用 ElGamal 椭圆曲线密码系统实现了一个安全的在线投票系统, 利用椭圆曲线密码体制的加性同态特性, 不需要对每一张选票进行单独解密从而保证了选票的安全性。区块链上的信息交换、价值转移和资产管理等重要操作由智能合约进行执行。智能合约是一种去中心化、去信任、可编程、不可篡改、自动执行合约条款的计算机交易协议[10], 将全同态加密技术融入到可编程的智能合约中, 对区块链内部数据安全风险管控能力将有较好提升。

2 相关工作

边缘计算模式为区块链提供了新的节点部署选择。文献[11]针对边缘计算中存在的问题, 提出了一种基于区块链的可信数据管理方案(BlockTDM), 该方案具有灵活可配置的体系结构。文献[12]提出了一种基于边缘计算模式的分层轻量级区块链框架(LLBF)和一种轻量级共识算法与动态信任权算法, 以提高区块链的吞吐量, 减少在新块中的验证计算量。文献[13]提出了一种基于车辆节点边缘计算模式的区块链 P2P 数据共享系统, 该系统采用基于信誉度的主观逻辑(TWSL)数据共享方案来保证数据的有效性。文献[14]提出了基于区块链的移动边缘计算应用框架, 该框架使用区块链和分布式安全数据库, 使诊疗方案操作性更强。文献[15]提出了基于边缘计算分层体系结构和区块链的 P2P 分布式账本, 使区块链与网络应用层分离, 降低了物联网设备的存储负担, 使用 Zerocash 和环签名保障了匿名性。文献[16]提出了一种智能电网区块链边缘计算模型(PBEM-SGN), 使用群签名和秘密通道授权技术来保证用户的合法性。文献[17]提出了在边缘环境中部署的带有隐私保护功能的区块链系统, 较好地解决了延迟和隐私保护问题。文献[18]提出了一种基于边缘计算的区块链门限环签名方案, 该方案可以保护数据所有者的隐私。文献[19]提出了一种基于边缘计算的 IIoT 身份管理和访问控制机制, 其中基于 Bloom filter 的访问控制机制提供了较好的安全保障, 基于自认证公钥的轻量级密钥协商协议提高了密钥协商的效率。文献[20]提出了一种区块链边缘计算模式下基于公钥和数字签名相结合的用户隐私保护方案。该方案抗攻击能力较强、边缘设备计算资源消耗较少。

智能合约为区块链提供了预置规则和更广泛的应用范围。文献[21]提出了一种基于以太坊智能合约的分散式身份存储系统, 用户的身份以属性方式进行存储和加密, 每个服务只能读取为其公开的标识属性, 而不能读取用户希望保密的属性。文献[22]设计了一种利用生物特征和区块链/智能合约来实现隐私保护的身份管理新方法, 其基本思想是使用区块链存储权威认证信息与个人生物特征识别转换值, 区块链上存储的数据由事先定义好的带有各种访问控制策略的智能合约来控制。文献[23]提出了基于以太坊的边缘计算智能合约 SmartEdge, 可以较好解决计算资源限制问题。文献[24]提出了一种基于动态访问控制来保护智能合约的框架, 该框架能提供对用户的隐私保护。文献[25]提出了一种基于零知识证明的 FabZK 联盟链系统, 该系统具有较好的隐私保护功能, 与 zk SNARKs、native Fabric 和 zkleger 相比该系统具有较好的性能。

同态加密技术为区块链的隐私保护提供了新方案。文献[26]提出了一个由区块链和同态加密支持的分布式智能电表数据聚合框架, 该框架能提高仪表

数据采集的安全性和保密性。文献[27]提出了一种用于敏感数据隐私保护的同态联盟区块链(HCB-SDPP), 设计了攻击实验来测试该系统的不同节点, 实验表明该系统安全性较好。文献[28]提出了基于同态加密和零知识证明的可追溯许可区块链, 其能保护交易隐私和交易的有效性。

本文的主要贡献如下:

(1) 提出了一种边缘计算模式下全同态加密的智能合约。采用边缘计算模式, 使 Hyperledger Fabric 系统较大提升了存储空间和计算效率。该智能合约通过调用 SEAL 库(Simple Encrypted Arithmetic Library)对 Fabric 网络中的数据库信息和边缘节点身份信息进行同态加密, 有效保护了区块链中交易信息的安全和边缘节点的身份隐私。

(2) 在 Hyperledger Fabric 系统中对基于 SEAL 库的全同态加密智能合约进行了运行和测试。经过测试和分析, 在 14 个边缘节点和 4 个 Fabric 网络通道组织成员节点同时访问系统时, 单节点的平均访问时间仅为 139.68 ms。与不采用边缘计算模式的系统相比, 在边缘模式下采用全同态加密智能合约的系统效率提升了 50%。相比 Kang 等人在 2019 年设计的目前性能最优的 FabZK 加密联盟链系统, 平均用时减少 35.84%, 效率提升 55.86%。本系统加密算法基于 BFV 同态加密方案, 具有较高的安全性。

3 预备知识

3.1 边缘计算

边缘计算主要由卡内基梅隆大学(CMU)于 2015 年发起, 该大学从学术研究的角度提出了具有广泛描述性意义的计算模型[28]。边缘计算采用网络、计算、存储、应用核心能力为一体的开放平台, 就近提供最近端服务[29]。其应用程序在边缘侧发起, 产生更快的网络服务响应, 满足行业在实时业务、应用智能、安全与隐私保护等方面的基本需求[30]。边缘计算处于物理实体和工业连接之间, 或处于物理实体的顶端。执行云端计算的实体也可以访问边缘计算的历史数据。

边缘计算的结构如图 1 所示, 其体系结构大致可分为三个层次, 即云端、中心网络、边缘端及其存储服务器。其中边缘存储服务器比云端服务器更接近边缘端。

云端有着比边缘端更强大的计算处理能力, 但随着海量边缘数据的爆炸式增长, 云端也产生了延迟增高、带宽不足、能耗过大等问题, 数据和信息的隐私保护风险也在加剧[31]。边缘计算的核心概念是

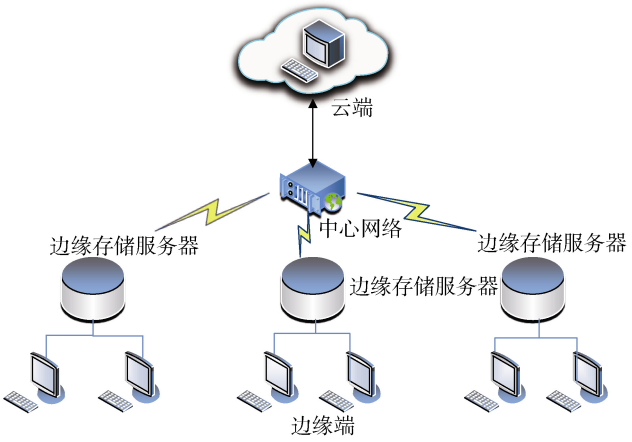

图 1 边缘计算模式结构图

Figure 1 Structure of edge computing mode

计算应该更接近于数据源, 也可以更接近用户。由于网络距离的减小, 诸如带宽、延迟和抖动等不稳定因素可以更容易地控制和改善。另外, 边缘网络的资源和用户处于相同的状态(如位置), 也可以根据场景信息为用户提供更方便的个性化服务(如基于位置的服务)。

作为部署在近端的边缘服务器, 不仅可以传输和分配流经网络的流量, 还可以满足实时数据处理、数据缓存和计算卸载等资源需求。因此, 为了获得更好的性能和更多的存储空间, 大多数计算和存储任务将在边缘端处理[32]。

3.2 智能合约

通过部署在共享账本上的智能合约, 用户可以维持自己的状态、控制自己的资产和对接收到的外界信息进行自动回应[33]。智能合约在可执行代码中定义了不同组织之间的规则, 应用程序通过调用智能合约来生成记录在分类账上的各种交易。使用区块链网络, 可以将这些合约转化为可执行程序并自动执行这些规则。

智能合约架构模型如图 2 所示。智能合约架构自上而下由应用层、智能运维层与合约层组成。

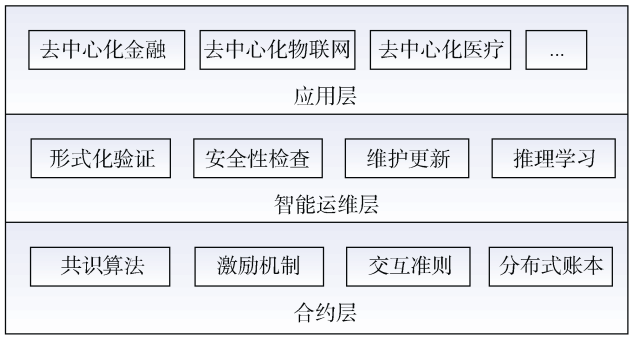

图 2 智能合约架构模型

Figure 2 Smart contract architecture model

应用层封装了智能合约在实际应用中的各种行业领域, 其中包括去中心化金融、去中心化物联网、去中心化医疗领域等等。智能运维层封装了一系列智能合约对静态合约数据的智能化动态操作, 包括形式化验证, 安全性检查, 维护更新, 推理学习等, 形式化验证和安全性检查是指利用精确的数学手段和强大的分析工具在合约的设计、开发测试过程中验证智能合约是否满足正确性和安全性, 以规范合约的生成和提高合约的可靠性和执行力。维护更新和推理学习是基于智能合约中的各类智能算法实现的, 智能运维层保证了智能合约能够按照设计者的意愿正确、安全、高效地运行。合约层封装了智能合约的静态合约数据, 包括共识算法、激励机制、交互准则和分布式账本。共识算法、激励机制和交互准则是智能合约交互和通信的基础, 分布式账本是智能合约执行完成后将执行结果记录的全节点共同维护的账本, 合约层代表了智能合约所有调用、执行、通信的规则。

目前的智能合约也存在不少安全风险[34]:

隐私泄露: 智能合约对区块链上的所有用户可见, 包括但不限于标记为 private 的资源, 存在造成隐私信息泄露的风险。

交易溢出与异常: 由于智能合约本身的约束条件, 如条件竞争、交易顺序依赖等, 可能会造成交易溢出与异常。

合约故障: 由于智能合约代码中可能存在不合理的故障处理机制, 从而导致异常行为。

拒绝服务: 由于各种原因导致的拒绝服务风险。

3.3 同态加密

同态加密是一种加密形式, 它允许人们对密文进行特定的代数运算, 解密后得到的结果与对明文的相同操作得到的结果一致[35]。

通常一个同态公钥加密方案包含 3 个算法: 密钥生成算法(KeyGen)、加密算法(Enc)和解密算法(Dec)。但是在同态加密中, 除了上述 3 个算法之外, 还包含第 4 个算法即密文计算算法(Eval)。

(1) 密钥生成算法 KeyGen(λ) 是一个随机化的算法, 输入安全系数 λ, 输出解密私钥 sk、加密公钥 pk 以及用于密文计算的公开密钥 ek。

(2) 加密算法 Enc(pk,m) 是一个随机化的算法, 输入公钥 pk 和明文 m, 输出密文 c, 对于相同的明文每次加密得到的密文都是不同的。

(3) 解密算法 Dec(sk,c) 是一个确定性算法, 输入私钥 sk 以及密文 c, 输出明文 m。

(4) 密文计算算法 Eval(ek,(c_1,c_2,$\cdots$,c_k),C), 输入密文计算密钥 ek, 电路 $C(c_1,c_2,\cdots,c_k)\in C_\lambda$ 和密文 (c_1,c_2,$\cdots$,c_k), 输出为密文计算结果 c^*。

若加密方案 a 对于加法 (+) 和乘法 (×) 都满足同态性, 且能在无限次运算后依然保持同态性, 则称该方案为全同态加密方案。

3.4 Hyperledger Fabric

Hyperledger Fabric 是联盟区块链的基本框架之一。Hyperledger Fabric 不同于过去的比特币和以太坊, 三者对比如表 1 所示。其使用模块化架构开发应用程序或解决方案, 实现了代码与分布式账本的分离。

表 1 三种区块链对比

Table 1 Comparison of three blockchains

	比特币	以太坊	超级账本
应用	加密货币	应用平台	应用平台
节能	否	否	是
智能合约执行环境	比特币虚拟机(BVM)	以太坊虚拟机(EVM)	Docker
编程语言	脚本语言	Solidity	Golang
货币	BTC	Ether	无
共识算法	POW	POW	Raft

对比比特币与以太坊, Hyperledger Fabric 的优势在于无需挖矿及相应的电能损耗, 不存在区块链的货币创建。Hyperledger Fabric 这种许可制的联盟链通常会使用数字证书等安全机制来增强安全性, 因此存在恶意节点的可能性很小。采用分布式崩溃故障容错共识算法 Raft, 降低了复杂性和成本。Raft 共识算法可以保证在系统中部分节点出现非拜占庭故障的情况下, 系统依然可以处理客户端的请求。

Raft 首先选举出主导节点(Leader Node), 其余节点启动后进入跟随状态。之后客户端向主导节点发送包含命令的请求。主导节点将收到的请求追加到其日志中, 并将该请求发送给所有的跟随节点(Follower Node)。跟随节点也会将该请求追加到自身的日志中并返回一个确认消息。一旦主导节点收到大部分跟随节点的确认消息, 就会将命令日志提交给其管理的状态机。一旦主导节点提交了日志, 跟随节点也会将日志提交给自身管理的状态机。最后主导节点更新区块向客户端返回响应结果。

4 智能合约架构设计

边缘计算模式下同态加密的智能合约架构如

图 3 所示, 其架构主要分为三层: 边缘层、智能合约层和网络层。

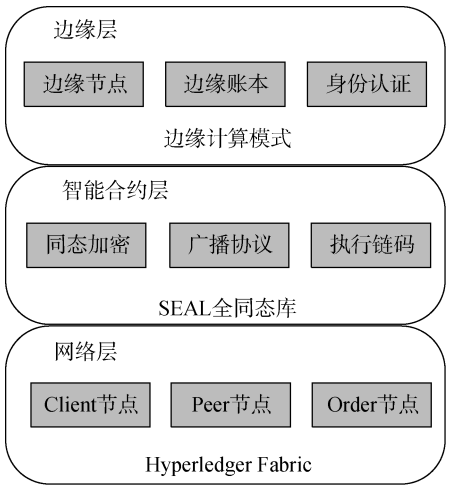

图 3　边缘计算模式下全同态加密智能合约架构

Figure 3　Architecture of fully homomorphic encryption smart contract in edge computing mode

其中网络层是整个系统的核心层, 网络层采用联盟链 Hyperledger Fabric, 其中包括 Client 节点、Peer 节点、Orderer 节点。Client 代表由最终用户操作的实体, 它需要连接到某一个 Peer 节点或者 Order 节点上与区块链网络通信。Peer 节点主要功能为记账、排序等。Orderer 负责接收包含签名的交易, 对未打包的交易进行排序生成区块, 广播给 peer 节点。智能合约层负责整个系统中网络应用端的安全以及节点之间交易的执行, 其主要功能包括基于 SEAL 库的全同态加密, 执行相关链码, 节点广播协议等。边缘层设置边缘节点和边缘账本, 其主要负责分担网络层 Hyperledger Fabric 的通道内的部分数据存储和计算功能, 在边缘层加入了身份隐私保护机制, 边缘节点在访问通道内交易数据及交易内容时, 访问者的身份采用全同态加密, 也就是说访问者的身份和 Fabric 网络中的交易数据和内容都可以实现隐私保护。

边缘模式下全同态加密智能合约的部署图如图 4 所示。首先用户端通过 Fabric-SDK 实现与 Fabric 的 WEB 连接, 通过 Fabric-CA 节点授予的证书认证后可以访问 Fabric 通道。如图 4 所示, 每个 Fabric 通道包含了 4 个加入的组织, 每个通道有一个独立的共享账本和一个独立的智能合约。

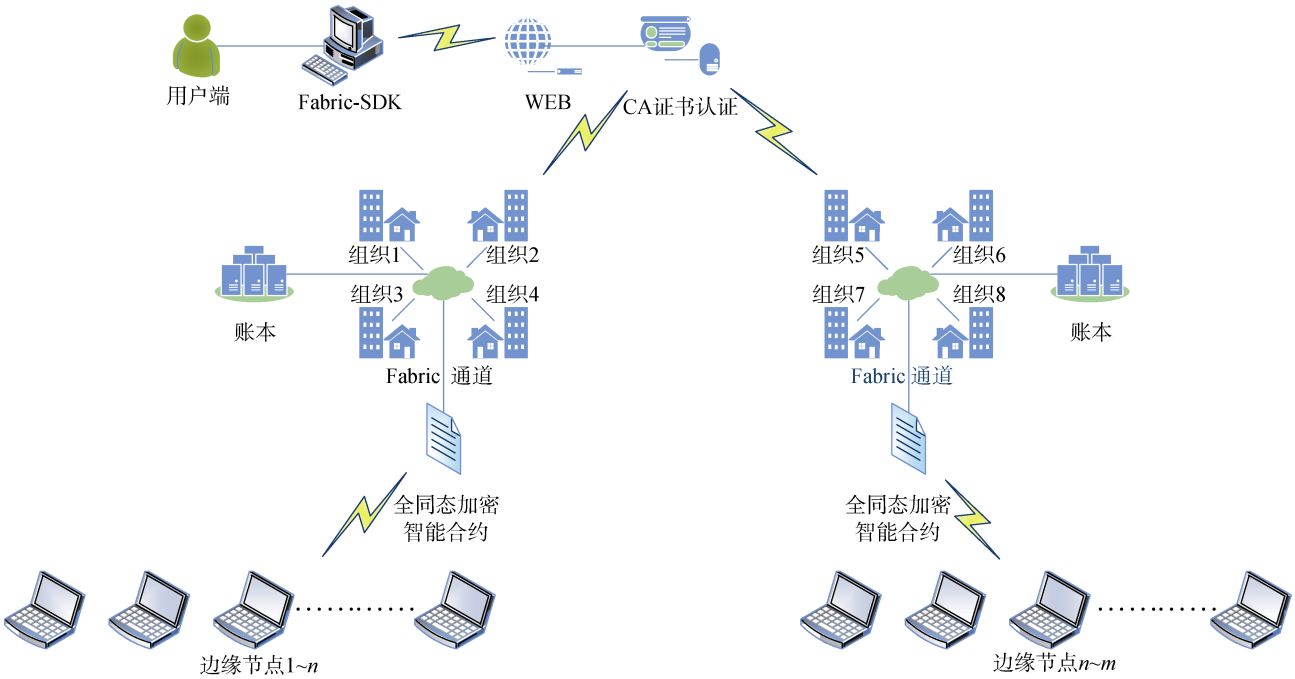

图 4　边缘模式下全同态加密智能合约部署图

Figure 4　Deployment of fully homomorphic encryption smart contract in edge mode

智能合约由 Fabric 通道内组织或者 Fabric 网络运营商制定。本文对每个通道制定相同的全同态加密智能合约, 该智能合约在 Fabric 通道内成员访问通道内账本数据时, 不需执行加密功能, 只执行基本的链码操作; 而在边缘节点访问 Fabric 通道时, 边缘节点的身份信息会执行全同态加密, Fabric 通道内的节点无法获悉其真实身份, 也就保护了访问者的身份隐私。另一方面, 对于边缘节点访问的 Fabric 通道内的账本重要数据也进行了全同态加密, 保障了通道内组织与组织之间重要信息的安全。采用边缘

计算模式, 降低了 Fabric 通道内数据传输的时延, 减轻了通道内账本的数据负担。

边缘模式下全同态加密智能合约有 4 个安全机制, 如下所述。

(1) 通过通道机制来实现联盟成员节点之间的私密通信。在联盟之下若干不同的组织建立了一个一个的通道, 每个通道都有一个独立的账本, 只有通道成员组织之间才能共享账本, 而其他组织无权访问账本。通道机制可以保证在成员组织之间形成一个专有的私密网络, 交易在其上以保密方式执行, 而与外部的无关组织或个人隔离开来。通道为联盟成员之间的私有通信和私有数据共享提供了一种机制, 通道保证了组织数据和通信的隐私。

(2) 通过基于 ECDSA 的 X.509 证书验证成员身份。ECDSA 的安全性基于椭圆曲线离散对数问题, 安全强度高于传统离散对数系统, 其具有较短的密钥、较小的计算参数、较快的计算和签名速度。成员的服务提供商(Membership Service Provider, MSP)是在 Fabric 信任的机构。更具体地说, MSP 是定义用于管理该组织的有效身份的规则的组件。MSP 实现使用X.509证书作为身份, 并采用传统的公共密钥基础结构(PKI)层次模型。证书的签名验签基于 ECDSA 和 SHA256 加密算法。通道内只有具有相同 MSP 证书的成员才能使用 Goosip 协议进行签名验签。

(3) 通过全同态加密机制, 实现边缘节点访问通道数据时的通道数据与节点身份的隐私保护。如图 4 所示, 组织1~4 处于 Fabric 通道内可以直接查看核心数据, 而边缘节点 1~n 想要访问通道内的数据, 需要经过智能合约, 智能合约对边缘节点执行身份认证后, 通过调用 SEAL 同态库, 对通道内的数据进行 BFV 全同态加密后传输给边缘节点。而边缘节点的身份经过智能合约的全同态加密后, 确保了边缘节点的身份隐私。

(4) 通过边缘计算模式, 实现分散存储, 减少了通道内节点的数量, 保障了通道内网络的稳定。通道外的节点作为边缘节点, 有固定的本地存储空间, 可以拓展 Hyperledger Fabric 网络的数据存储量。边缘节点访问本地数据库时无需共识算法验证与数据同步传输。在 Fabric 通道内部的数据为核心数据, 利用边缘计算减轻了通道传输的网络负担。

5 基于 SEAL 库的智能合约设计

5.1 基于 BFV 算法方案的 SEAL 库

Fan 等人于 2012 年提出了 BFV 同态方案, 该方案可视为对 Brakerski 所提方案的优化。BFV 全同态加密方案共包括私钥生成、公钥生成、计算密钥生成、加密、解密、加法和乘法等 7 个算法[36]。

近年来微软不断对基于 BFV 方案的同态加密库 SEAL 进行优化和改进。2017 年微软发布 SEAL2.3.0, 使其可以支持 Bajard 等人的方案。2018 年微软发布 SEAL3.0, 使其支持 Cheon 等人提出的 CKKS 方案, 可以实现实数域上的密文近似计算。2019 年发布 SEAL3.2, 增加了对.NET 开发的完整支持, 使.NET 开发人员编写同态加密应用程序更为便捷。目前 SEAL 的版本为 SEAL3.4。SEAL 库的安全性基于 RLWE 格困难问题, 理论上可抵御量子攻击。

SEAL 库中主要有如下几个较为常用的函数: set_poly_modulus_degree() 、 set_coeff_modulus() 和 set_plain_modulus()是 3 个用于参数设置的函数; KeyGenerator 类下的 public_key()和 secret_key()用于生成公私钥; 在进行密文运算时, 主要用到 Evaluator 类下的相关函数, 例如 square(a, b)为平方运算函数, add_plain_inplace(a, b)为加运算函数, multiply(a, b, c)为密文乘运算函数等。此外若要进行重线性化, 需要使用 KeyGenerator::relin_keys()来生成重线性化密钥; 然后在每次乘法运算后使用 Evaluator::relinearize_inplace()函数对密文进行重线性化, 重线性化可以有效提高全同态运算的效率。

5.2 智能合约设计

在 Hyperledger Fabric 系统中, 智能合约运行在一个受保护的 Docker 容器中, 与节点的运行相互隔离。通常智能合约需要通过 gPRC 协议与 Peer 节点进行交互通信。在边缘端智能合约的执行过程如图 5 所示。

(1) 边缘客户端向 Fabric 的边缘节点 Peer 发起请求。边缘客户端通过 Fabric-SDK 首先进行身份的验证, 验证内容为 Fabric 在网络建立时设置的验证方式, 通常包括 CA 证书验证、用户账号/口令验证、系统设置的权限验证等。验证完成后, 访问请求将发送至边缘节点 Peer。

(2) 边缘节点收到客户端发来的请求后, 向 Docker 容器中的智能合约发送包含相关需求的请求, 并执行智能合约操作。在启动智能合约前, 所有指定的边缘节点接收到请求, 并通过数字签名验证来判断请求的合法性。验证请求合法后才会启动智能合约。

(3) 智能合约执行预先设定好的合约流程, 对状态数据库执行数据操作。智能合约访问通道账本, 执行 getState()函数, 对账本执行读取操作。

(4) 状态数据库收到智能合约指令后, 调取其状态数据并返回给智能合约, 状态数据处于不可篡改的状态。

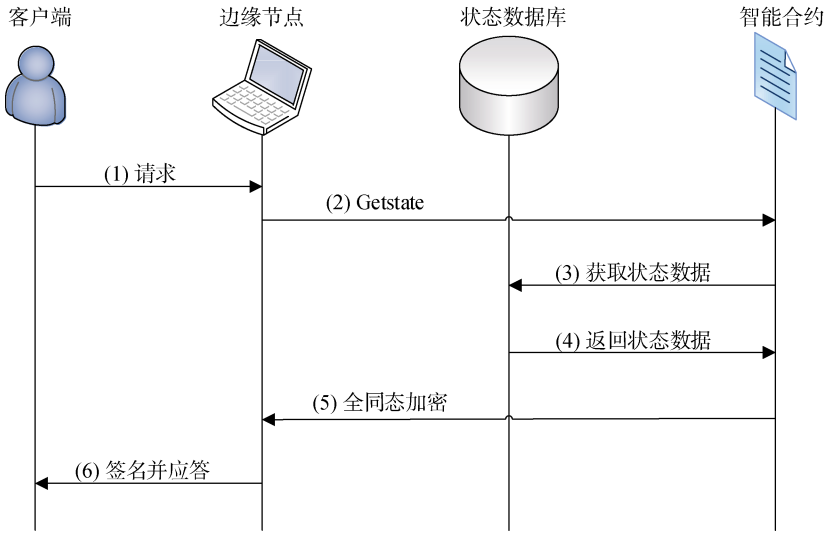

图 5　边缘端智能合约执行过程

Figure 5　Execution process of smart contract in edge mode

(5) 智能合约收到状态数据库的读集合后, 执行表 2 的合约操作并对状态数据进行全同态加密, 之后返回加密数据给边缘节点 Peer。

(6) 在完成智能合约操作后, 边缘节点将加密数据存于本地并执行数字签名后返回给客户端。

我们在智能合约中设计了交易信息的数据结构如表 2 所示。

表 2　智能合约数据结构

Table 2　Data structure of smart contract

Data Structure smart contract
type Transaction struct {
ObjectType　string `json:"docType"`
IdentityA　string　`json:"IdentityA"`
IdentityB　string　`json:"IdentityB"`
OrganizationA　string　`json:"OrganizationA"`
OrganizationB　string　`json:"OrganizationB"`
Single transaction amount string `json:"Single transaction amount"`
Gmv　string `json:"Gmv"`
Transaction content　string　`json:"Transaction content"`
}

智能合约数据结构包括付款方的身份 IdentityA、付款方所属组织 OrganizationA、收款方所属组织 OrganizationB、单次交易金额 Single transaction amount、交易总额 Gmv、交易内容 Transaction content 等。

当同一时间通道内查询用户过多时, 过多的交易信息数据将会降低通道内网络的稳定性。为分担 Fabric 通道内的数据存储和网络负载, 提高查询效率, 将大部分非组织内成员的节点划归为边缘节点。智能合约将为用户设置不同的安全方案, 通道内的组织用户可查询所有的交易信息及历史信息, 而边缘节点用户只能查询全同态加密后的当前状态的交易内容。如下表 3 智能合约算法 1 中描述了智能合约在查询过程中如何划分边缘节点并对交易内容执行全同态加密。

表 3　智能合约算法 1

Table 3　Smart contract algorithm 1

Algorithm 1 Smart contract based on BFV homomorphic encryption
Initialization check if user's label is Edge node
if The label is Edge node **then**
1: **Input**: string “key”
2: Use function GetState to query date and pass in parameter key
3: json.Unmarshal for key → a
4: SHA256 hash operation on a → A
5: SHA256 hash operation on the date from CouchDB → B
6: comparison of SHA256 Hash Results A and B
7: A=B Indicates that the data has not been modified
8: Link SEAL Library→BFV Encryption algorithm function
9: QueryResultsIterator(stringkey)→Correspondingall BFVencfun(data)
else if The label is core member **then**
10: **Input**: string key
11: QueryResultsIterator(string key)→Corresponding all data
12: **end if**
13: **return** key;

首先, 初始化检查登录用户的标签是否为边缘节点, 如果是, 智能合约通过关键字 key 查询相关的交易信息。其次执行 json.Unmarshal 来处理输入数

据获得 a, 通过摘要运算获得哈希结果 A。然后从 CouchDB 中获取数据, 并执行 SHA256 操作来获取散列结果 B。接着判断散列结果 A 和 B 是否相等, 如果它们相等, 则表示数据未被修改, 然后合并来自区块链和 CouchDB 的数据, 再调用 SEAL 库对相关数据进行全同态加密, 通过输入的关键字 key 查询到对应的一组全同态加密后的交易信息 BFVencfun (data)。

如果初始检查的用户标签是通道组织内的成员, 则能查询到当前状态交易内容的明文。SHA256 算法可以为任何长度的数据生成一个 256 位长的哈希值, 该哈希值可以验证数据的完整性。下表 4 为本文使用的 SHA256 算法, 表 5 为调用 SEAL 库中的 BFV 全同态加密算法。

表 4 SHA256 算法

Table 4 SHA256 algorithm

Algorithm 1 SHA256

1: Initialization parameters

Get 8 parameters H_0~H_7

2: Message list: $W_t = M_t^i (0 \leqslant t \leqslant 15)$

$W_t = \sigma_1^{256}(W_{t-2}) + W_{t-7} + \sigma_0^{256}(W_{t-15}) + W_{t-16} (16 \leqslant t \leqslant 63)$

3: 每轮哈希值的中间结果初始化 8 个变量

4: 执行压缩函数

$T_1 = h + \sum_1^{256}(e) + Ch(e,f,g) + K_t^{256} + W_t (0 \leqslant t \leqslant 63)$

$T_2 = h + \sum_0^{256}(a) + M_{aj}(a,b,c)$

$h = g; g = f; f = e; e = d + T_1; d = c; c = b;$

$b = a; a = T_1 + T_2;$

5: 将压缩区块添加到当前哈希值

$H_0^i = a + H_0^{i-1}, H_1^i = b + H_1^{i-1},$

$H_2^i = c + H_2^{i-1}, H_3^i = d + H_3^{i-1},$

$H_4^i = e + H_4^{i-1}, H_5^i = f + H_5^{i-1},$

$H_6^i = g + H_6^{i-1}, H_7^i = h + H_7^{i-1}.$

边缘节点访问 Fabric 通道内账本数据时, 其身份信息通过同态加密, 相应的身份隐私保护算法如下表 6 中的智能合约算法 2 所示。

边缘节点对于本身产生的交易数据将存储在本地数据库中, 当边缘节点访问组织成员在 Fabric 网络中的通道时, 需要由智能合约进行身份认证和加密。首先智能合约验证边缘节点的相关信息, 包括标签 label, 身份 ID, 地址 address。当依次验证成功后, 对边缘节点的身份信息进行同态加密, 调用部署好的 SEAL 库, 利用函数 BFVencfun()对身份信息 Identity(label, ID, address)加密, 完成后将此身份信息链接到 Fabric Channel, 如果身份验证不通过, 则

表 5 BFV 全同态算法

Table 5 BFV fully homomorphic encryption algorithm

Algorithm 2 BFV → BFVencfun()

1: SecretKeyGen(λ)

$s \xleftarrow{\$} R_2 \quad sk = s$

2: PublicKeyGen(sk)

$s = sk \quad a \xleftarrow{\$} R_q \quad e \leftarrow \chi$

$pk = ([-(as + e)]_q, a)$

3: EvaluationKeyGen(sk, ω)

$a_i \xleftarrow{\$} R_q \quad e_i \leftarrow \chi \quad i \in \{0, \cdots, \ell\}$

$evk = ([-(a_i s + e_i) + \omega^i s^2]_q, a_i)$

4: Encrypt(pk, m)

$m \in R_t \quad pk = (p_0, p_1) \quad e_1, e_2 \leftarrow \chi$

$ct = ([\Delta m + p_0 u + e_1]_q, [p_1 u + e_2]_q)$

5: Decrypt(sk, ct)

$s = sk \quad c_0 = ct[0] \quad c_1 = ct[1]$

output: $\left[\left\lfloor \frac{t}{q} [c_0 + c_1 s]_q \right\rceil \right]_t$

表 6 智能合约算法 2

Table 6 Smart contract algorithm 2

Algorithm 2 Smart contract based edge node authentication

Initialization Verify Identity(label ,ID, address)

if The label is Edge node **then**

1: Check Edge node(ID)

if Edge node(ID)=true **then**

2: Check Edge node(ID(address))

if Edge node(ID(address))=true **then**

3: Validation passed

4: Identity(label ,ID, address) encryption

5: Link SEAL Library→BFV Encryption algorithm function

6: Identity(label ,ID, address)→BFVencfun(Identity(label ,ID, address))

7: Link Fabric Channel

else if The label is not Edge node **then**

8: Check if the label is Channel member then

9: Link Fabric Channel

else if

10:Access failed

end if

return Identity(label ,ID, address);

检查是否为 Fabric Channel 中的组织成员, 若是组织成员则直接通过通道规则链接到通道内的账本。

6 正确性与安全性证明

下面从正确性和安全性两方面对方案进行证明。

6.1 正确性证明

(1) 身份验证的正确性。采用公共密钥基础结构(PKI)层次模型。通过 MSP 向成员分发证书, 证书的签名验签基于 ECDSA 和 SHA256 加密算法。通道内只有具有相同MSP证书的成员才能使用Goosip协议进行签名验签。

(2) 智能合约加/解密的正确性。通过证明以下引理的正确性, 来说明通过 BFV 全同态加密算法运算后得到的密文能够被正确解密。

首先介绍相关参数, 多项式环 $R=Z[x]/(f(x))$, 其中 $f(x)\in Z(x)$ 且 $f(x)$ 的指数为 d, R 的膨胀因子被定义为 $\delta_R=\max\{\|a\cdot b\|/(\|a\|\cdot\|b\|):a,b\in R\}$, v 为密文中包含的噪声, r 为线性化误差, r 的值很小, χ 为区间 $[-B,B]$ 上的整数分布。

其次简要说明一下 BFV 加解密算法。λ 为安全系数, ω 为对数的底数, 明文空间为 R_t, $t>1$。使 $\Delta=\lfloor q/t\rfloor$, $\gamma_t(q)=q \bmod t$ 得 $q=\Delta\cdot t+\gamma_t(q)$。$a\xleftarrow{\$}S$ 表示 a 是有限集合 S 的一个均匀抽样。

私钥生成: 均匀选取 $s\xleftarrow{\$}R_2$ 并输出 $sk=s$;

公钥生成: 输入 $s=sk$, 选取 $a\xleftarrow{\$}R_q$ 和 $e\leftarrow\chi$, 输出 $p_k=([-(a\cdot s+e)]_q,a)$。

加密算法: 明文 $m\in R_t$, 将公钥表示为 $pk=(p_0,p_1)$。作抽样 $e_1,e_2\leftarrow\chi$, 计算 $ct=([\Delta m+p_0u+e_1]_q,[p_1u+e_2]_q)$。

解密算法: 令 $s=sk$, $c_0=ct[0]$, $c_1=ct[1]$。输出 $\left[\lfloor\frac{t}{q}[c_0+c_1s]_q\rceil\right]_t$。

引理: 使用以上 BFV 算法的相关条件并假设 $\|\chi\|<B$,可得

$$[c_0+c_1\cdot s]q=\Delta\cdot m+v$$

且 $\|v\|\leqslant 2\cdot\delta_R\cdot B^2+B$。表明当 $2\cdot\delta_R\cdot B^2+B<\Delta/2$ 时, $BFV.Dec_s(c)=\lfloor m\rfloor_t$ 成立。

证明: 模 q 给出的定义为 $c_0+c_1\cdot s=p_0\cdot u+e_1+\Delta\cdot m+p_1\cdot u\cdot s+e_2\cdot s \bmod q$ 由于 $p_k=(p_0,p_1)=([-(a\cdot s+e)]_q,a)$, 将 p_0,p_1 代入上式, 可得:

$$c_0+c_1\cdot s=\Delta\cdot m+e\cdot u+e_1+e_2\cdot s \bmod q$$

因为 $e\cdot u+e_1+e_2\cdot s$ 在 R_q 上为足够小的误差项, 可得出噪声 $v=e\cdot u+e_1+e_2\cdot s$。因为 $e,e_1,e_2,u,s\leftarrow\chi$, 可以得出给定的界 $\|v\|\leqslant 2\cdot\delta_R\cdot B^2+B$。

所以, $c_0+c_1\cdot s=\Delta\cdot m+v+r\cdot q$, 对该式除以 q, 乘以 t, 得到 $m+(t/q)\cdot(v-\varepsilon\cdot m)+t\cdot r$, 其中 $\varepsilon=q/t-\Delta=\gamma_t(q)/t<1$。为确保正确解密, 依据明文空间 R_t 大小, 设置参数 t、q 取值, 使得 $(t/q)\cdot\|v-\varepsilon\cdot m\|<1/2, m\in R_t$ 即可。

证毕。

6.2 安全性证明

(1) 身份验证的安全性。如果恶意攻击者试图伪造或窃取节点的身份, 他就必须伪造由 Fabric 系统 CA 或是权威机构生成的基于 ECDSA 算法的签名证书, 这在计算上是不可行的, 也就是说 Fabric 中节点的身份是可信的。

(2) 智能合约加/解密的安全性。由于智能合约加/解密协议的安全性可以规约为全同态加密算法的安全性。所以, 这里通过不可区分选择明文攻击(IND-CPA)模型来证明 BFV 全同态加密的安全性。

初始化阶段: 挑战者运行公钥生成算法(PublicKeyGen(sk)): 输入 $s=sk$, 选取 $a\xleftarrow{\$}R_q$ 和 $e\leftarrow\chi$, 输出 $pk=([-(as+e)]_q,a)$,将 pk 交予敌手 A。

预言机访问阶段: 敌手 A 选择明文 $m_i,i=0\cdots n$, 询问加密预言机, 在得到密文应答之后, 敌手A多次分阶段提交选择的明文 $m_i,i=0\cdots n$, 并得到对应于不同明文的密文 $c_i,i=0\cdots n$。

挑战阶段: 敌手 A 选取两个明文 m_0 和 m_1, 把这两个明文发送给挑战者, 挑战者选取 $b\in\{0,1\}$, 并将挑战密文 $ct=\left([\Delta m_b+p_0u+e_1]_q,[p_1u+e_2]_q\right)$ 发送给敌手 A。

猜测阶段: 敌手 A 输出 b', $b'=b$ 则攻击成功。由于该同态加密方案基于 RLWE 问题, 所以敌手 A 的优势可以规约为解决 RLWE 问题的优势, 故

$$Adv_{BFV,A}(\lambda)=\left|\Pr[b=b']-1/2\right|=RLWE_{d,q,\chi}(A)$$

也就是说, 敌手A的优势基本可忽略不计, 因此对于任意多项式时间的敌手 A, 在游戏中获胜的概率 $Adv_{IND-CPA}(A)$ 满足:

$$\left|Adv_{IND-CPA}(A)-1/2\right|=negl(\lambda)$$

因此, 该算法是 IND-CPA 安全的。

7 测试与分析

我们开发了基于 Hyperledger Fabric 的原型系统,

以验证提出的边缘模式下全同态加密的智能合约的有效性。该系统在 ubuntu16.04(64 位)虚拟机, Intel®CoreTMit-4700HQ CPU @2.5Ghz 处理器和 8G 内存上运行。

在 SEAL 库中不同参数的选择将影响全同态加密算法的运算效率及安全性[36]。安全参数 poly_modulus_degree 代表分圆多项式的次数, 它的值取 2 的整数次幂。poly_modulus_degree 取值越大, 则方案越安全, 但也会使密文体积变大, 导致初始化、加密、同态加、同态乘、解密等操作效率降低。明文空间参数 plain_modulus 可以设置为任意的正整数, 其不仅决定着明文数据的大小, 同时也会影响新鲜密文的噪声以及同态乘运算后的噪声增长。根据表 7 安全参数选择测试和表 8 明文空间选择测试的结果, 我们的实验参数选取如下: poly_ modulus_degree(2048)、coeff_modulus (54bit)、plain_modulus(256)。

表 7 安全参数选择测试

Table 7 Test for key parameters

安全参数 poly_modulus_degree	初始化时间 (ms)	加密时间 (ms)	加法时间 (ms)	乘法时间 (ms)	加法解密时间 (ms)	乘法解密时间 (ms)
2048	14.42	2.38	—	1.36	0.2	0.16
4096	72.16	5.48	—	4.7	0.5	0.66
8192	424.2	14.94	0.18	17.64	1.54	2.48
16384	2760.88	44.78	1.08	71.62	6.32	9.84

表 8 明文空间选择测试

Table 8 Test for plaintext space

明文空间 plain_modulus	初始化时间 (ms)	加密时间 (ms)	加法时间 (ms)	乘法时间 (ms)	加法解密时间 (ms)	乘法解密时间 (ms)
101	0.01462	0.00242	2.00E-05	0.00148	0.00018	0.0001
5101	0.01442	0.00238	—	0.00136	0.0002	0.00016
15101	0.01432	0.00228	—	0.0015	0.00012	0.00016
20101	0.01424	0.00242	—	0.00132	0.00016	0.0002

边缘端节点实现跨通道查询交易的功能主要在于其拥有本地和核心通道内的链码和账本, 本地链码和账本采用系统原生的加密及运行方式。在本地节点实现的交易数据存储于本地, 具有较高的效率; 而边缘节点对于核心通道内账本由于受限于其链码的设置, 其仅具备读功能。对边缘节点的配置信息如下:

(1) 通信传输协议: gossip 协议

(2) 身份验证证书: MSP_{local}、$MSP_{channel}$

(3) 账本: $Ledger_{local}$、$Ledger_{channel}$

(4) 链码: $Chaincode_{local}$、$Chaincode_{channel}$

(5) 存储: $Ledger_{local}$

(6) 写功能: $Ledger_{local}$

(7) 读功能: $Ledger_{local}$、$Ledger_{channel}$

(8) 权限设置: $Chaincode_{channel}$

我们对采用边缘计算模式和不采用边缘计算模式的原型系统进行了测试。采用边缘计算模式时, 该系统存储 600 个交易信息数据, 通道内有 4 个组织成员节点, 其中 100 条交易信息存储在 Fabric 通道账本内, 500 条交易信息数据存储在边缘端。边缘端节点数量从 2 个开始依次增加到 4、6、8、10、12、14, 然后由所有节点共同发起对原型系统的随机访问。表 9 显示了边缘节点数量和访问所需时间的关系。访问总时间是所有节点在访问开始到访问成功所需的时间; 单节点访问时间是指访问总时间除以查询节点总数。

表 9 边缘计算模式下边缘节点数量和访问时间的关系

Table 9 Relationship between the number of edge nodes and access time in edge computing mode

边缘节点数量	通道内节点数量	访问总时间(ms)	单节点访问时间(ms)
2	4	916.38	152.73
4	4	1212.64	151.58
6	4	1498.52	149.85
8	4	1766.76	147.23
10	4	2025.38	144.67
12	4	2264.64	141.54
14	4	2514.24	139.68

在不采用边缘计算模式时, 在 Fabric 通道内存储 600 条交易信息数据, 在 Fabric 通道内依次设置了 6、8、10、12、14、16、18 个 Peer 节点。表 10 显

示了非边缘模式下节点数量和访问时间的关系。

表 10　非边缘计算模式下不同节点数和访问时间的关系

Table 10　Relationship between the number of different nodes and access time in non edge computing mode

边缘节点数量	通道内节点数量	访问总时间(ms)	单节点访问时间(ms)
0	6	944.70	157.45
0	8	1270.24	158.78
0	10	1609.71	160.97
0	12	2005.92	167.16
0	14	2490.46	177.89
0	16	3044.08	190.25
0	18	3759.66	208.87

图 6 显示了边缘计算模式(下方曲线)和非边缘计算模式(上方曲线)下, 随着节点数量增加访问总时间变化的曲线图。从图 6 可以看出随着节点数量增加, 访问总耗时也增加, 非边缘计算模式下的访问总耗时要高于边缘计算模式, 这是由于边缘节点在访问 Fabric 通道数据库时, 智能合约会对数据以及边缘节点的身份进行同态加密。而随着边缘节点数量的递增, 边缘计算模式的优势逐渐体现出来。在非边缘模式下 18 个节点进行数据访问操作时, 计算机 CPU 利用率达到了 90%以上, 系统内存占比达 73%。边缘计算模式下, 单节点的访问时间随着边缘端节点的增多而显著降低, 在边缘节点为 14 时, 稳定情况下计算机 CPU 利用率在 70%左右, 系统内存占比仅为 56%。这是因为在边缘计算模式下, 边缘节点将边缘端发生的交易数据信息存储在本地数据库, 大大扩展了整个系统的存储空间。在本地数据库访问时无需共识算法验证、X.509 证书身份验证以及数据同步的处理, 且区块链网络上的数据量也明显减少, 降低了区块链计算的数据流量和存储负载。故而随着边缘节点数目的增多, 边缘计算模式的优势更为明显, 两种模式的对比区别也愈发明显。

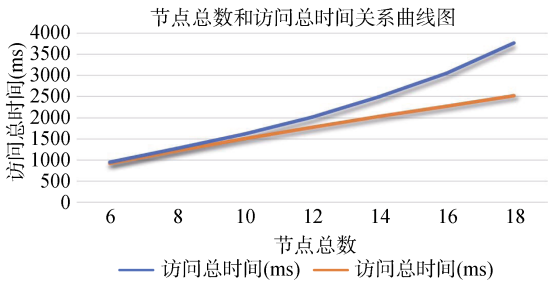

图 6　节点总数和访问总时间关系曲线图

Figure 6　Relationship between the total number of nodes and the total access time

图 7 为边缘计算模式(下方曲线)和非边缘计算模式(上方曲线)下节点数量与单节点访问时间的对比曲线图。当节点总数高于 10 时, 非边缘计算模式的 Hyperledger Fabric 网络需要承载更多的计算数据量和更大的数据库时, 其单节点访问时间显著增加, 网络出现延迟, 计算资源产生不足。在节点总数为 18, 边缘节点为 14 的情况下, 系统的单节点访问时间仅为 139.68 ms, 而非边缘计算模式下, 单节点访问时间达到 208.87 ms, 也就是说, 采用边缘计算模式比采用非边缘计算模式的效率提升了 50%。

本文选取目前较新的 4 个区块链系统进行性能对比, 如表 11 所示。文献[23]提出了基于以太坊平台的边缘计算智能合约, 该智能合约不涉及身份和数据的加密, 平均响应时间为 8600 ms。文献[22]设计实现了身份隐私保护的智能合约, 采用 SHA-256 执行

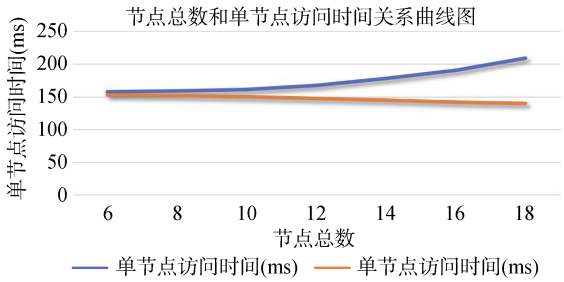

图 7　节点总数和单节点访问时间曲线图

Figure 7　Relationship between the total number of nodes and the access time of a single node

表 11　系统性能对比

Table 11　System performance comparison

	区块链平台	是否采用边缘计算模式	是否数据/身份隐私保护	加密方案	平均响应时间(ms)
文献[23]	以太坊	是	否	无	8600
文献[22]	以太坊	否	是	SHA-256	15000
文献[11]	Hyperledger fabric	是	是	自行设计	618
文献[25]	Hyperledger fabric	否	是	FabZK	217.7
本文	Hyperledger fabric	是	是	BFV	139.68

摘要运算, 没有采用边缘计算模式, 在以太坊平台进行测试, 平均响应时间为 15000 ms。文献[11]设计了基于区块链的边缘计算可信数据管理方案, 该方案具备身份认证、身份隐私与加密解密功能; 应用边缘计算模式, 在 Hyperledger Fabric 平台测试平均响应时间为 618 ms。文献[25]是基于零知识证明的 FabZK 联盟链系统, 该系统对身份信息进行加密, 在 Hyperledger Fabric 平台测试其平均响应时间为 217.7 ms。与这 4 个区块链系统中的最好结果[25]相比, 本文测试得到的平均响应时间为 139.68 ms, 响应时间减少了 35.84%, 效率提升 55.86%。实验表明, 我们的系统接入效率较好, 性能较为优良。

8 小结

本文通过将边缘计算模式应用于联盟链 Hyperledger Fabric 系统中, 提高了 Fabric 网络的计算效率, 降低了对存储空间的需求, 并在此基础上设计实现了边缘计算模式下的全同态加密智能合约。该智能合约通过调用 SEAL 库对 Fabric 的交易数据库信息和边缘端节点身份进行同态加密, 在保持交易数据可计算性的同时, 实现了交易信息和边缘节点身份的隐私保护。经过测试和分析, 在 14 个边缘节点同时访问原型系统时, 平均访问用时仅为 139.68 ms, 与目前较新的其他 4 个区块链系统相比, 平均响应时间最短。我们设计的边缘模式下的全同态加密智能合约可以有效提高 Hyperledger Fabric 系统中网络的运行效率, 同时减少了对存储空间的要求。

参考文献

[1] Ma C, Kong X, Lan Q, et al. The privacy protection mechanism of Hyperledger Fabric and its application in supply chain finance[J]. *Journal of cybersecurity science and technology*, 2019, 2(1): 49-51.

[2] Esposito C ,Castiglione A ,Pop F , et al. Challenges of connecting edge and cloud computing: A security and forensic perspective[J]. *IEEE Cloud Computing*, 2017, 4(2): 13-17.

[3] Xiong Z , Zhang Y , Niyato D , et al. When Mobile Blockchain Meets Edge Computing[J]. *IEEE Communications Magazine*, 2018, 56(8): 33-39.

[4] Liu M , Yu F R , Teng Y , et al. Distributed Resource Allocation in Blockchain-Based Video Streaming Systems With Mobile Edge Computing[J]. *IEEE Transaction on Wireless Communications*, 2019, 18(1): 695-708.

[5] Liu H , Zhang P , Pu G , et al. Blockchain Empowered Cooperative Authentication With Data Traceability in Vehicular Edge Computing[J]. *IEEE Transactions on Vehicular Technology*, 2020, 69(4): 4221-4232.

[6] Li Zengpeng, Ma Chunguang, Zhou Hongsheng. Overview on fully homomorphic encryption[J]. *Journal of Cryptologic Research*, 2017, 4(6): 561-578.
(李增鹏, 马春光, 周红生. 全同态加密研究[J]. *密码学报*, 2017, 4(6): 561-578.)

[7] Xiao L , Deng H , Tan M , et al. Insurance Block: A Blockchain Credit Transaction Authentication Scheme Based on Homomorphic Encryption[C]. *China: BlockSys*, 2019: 747-751.

[8] Marella P B , Milojkovic M , Mohler J , et al. GenVote: Blockchain-Based Customizable and Secure Voting Platform[C]. *Portugal: ICISSP*, 2018: 152-171.

[9] Chandra P J , Sathia B P R K , Swarnalaxmi S , et al. Blockchain Centered Homomorphic Encryption: A Secure Solution for E-Balloting[C]. *Proceeding of the International Conference on Computer Networks, Big Data and IoT. India: ICCBI*, 2018: 811-819.

[10] Worley C R, Skjellum A. Opportunities, Challenges, and Future Extensions for Smart-Contract Design Patterns[C]. *The Workshops on Business Information Systems. Germany: BIS*, 2018: 264-276.

[11] Ma Z F, Wang X C , Jain D K , et al. A Blockchain-Based Trusted Data Management Scheme in Edge Computing[J]. *IEEE Transactions on Industrial Informatics*, 2019, 16(3): 2013-2021.

[12] Xu X , Zeng Z , Yang S , et al. A Novel Blockchain Framework for Industrial IoT Edge Computing[J]. *Sensors*, 2020, 20(7): 2061.

[13] Kang J , Yu R , Huang X , et al. Blockchain for Secure and Efficient Data Sharing in Vehicular Edge Computing and Networks[J]. *IEEE Internet of Things Journal*, 2019, 6(3): 4660-4670.

[14] Rahman M A , Hossain M S , Loukas G , et al. Blockchain-based Mobile Edge Computing Framework for Secure Therapy Applications[J]. *IEEE Access*, 2018, 6: 72469-72478.

[15] Nyamtiga B W, Sicato J C S, Rathore S, et al. Blockchain-Based Secure Storage Management with Edge Computing for IoT[J]. *Electronics*, 2019, 8(8):828-850.

[16] Gai K, Wu Y, Zhu L , et al. Permissioned Blockchain and Edge Computing Empowered Privacy-Preserving Smart Grid Networks[J]. *IEEE Internet of Things Journal*, 2019, 6(5): 7992-8004.

[17] Jayasinghe U , Lee G M , Macdermott I , et al. TrustChain: A Privacy Preserving Blockchain with Edge Computing[J]. *Wireless Communications and Mobile Computing*, 2019, 2019(1): 1-17.

[18] Wang Z , Fan J. Flexible Threshold Ring Signature in Chronological Order for Privacy Protection in Edge Computing[J]. *IEEE Transactions on Cloud Computing*, 2020, 29(7): 4954-4965.

[19] Ren Y , Zhu F , Qi J , et al. Identity Management and Access Control Based on Blockchain under Edge Computing for the Industrial Internet of Things[J]. *Applied Sciences*, 2019, 9(10): 2058-2074.

[20] Ernest B, Shiguang J. Privacy Enhancement Scheme (PES) in a Blockchain-Edge Computing Environment[J]. *IEEE Access*, 2020, 8:25863-25876.

[21] Friebe S, Sobik I , Zitterbart M . DecentID: Decentralized and Privacy-Preserving Identity Storage System Using Smart Contracts[C]. *2018 17th IEEE International Conference on Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering. New York: TrustCom/BigDataSE*, 2018: 37-42.

[22] Liu Y , Sun G , Schuckers S . Enabling Secure and Privacy Preserving Identity Management via Smart Contract[C]. *2019 IEEE Conference on Communications and Network Security. Washington DC:CNS*, 2019: 1-8.

[23] Wright K L , Martinez M , Chadha U , et al. SmartEdge: A Smart Contract for Edge Computing[C]. *2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data. Halifax: IEEE*, 2018: 1685-1690.

[24] Gupta R , Shukla V K , Rao S S , et al. Enhancing Privacy through "Smart Contract" Using Blockchain-Based Dynamic Access Control[C]. *2020 International Conference on Computation, Automation and Knowledge Management. Dubai: ICCAKM*, 2020: 338-343.

[25] Kang H , Dai T , Jean-Louis N , et al. FabZK: Supporting Privacy-Preserving, Auditable Smart Contracts in Hyperledger Fabric[C]. *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. Portland: DSN*, 2019: 543-555.

[26] Wang Y , Luo F , Dong Z Y , et al. Distributed Meter Data Aggregation Framework Based on Blockchain and Homomorphic Encryption[J]. *IET Cyber Physical Systems Theory & Applications*, 2018, 4(1): 30-37.

[27] She W , Gu Z H , Lyu X K , et al. Homomorphic Consortium Blockchain for Smart Home System Sensitive Data Privacy Preserving[J]. *IEEE Access*, 2019, 7: 62058-62070.

[28] Mitani T , Otsuka A . Traceability in Permissioned Blockchain[C]. *2019 IEEE International Conference on Blockchain. Atlanta:Blockchain*, 2019: 286-293.

[29] Wang B , Li M , Jin X , et al. A Reliable IoT Edge Computing Trust Management Mechanism for Smart Cities[J]. *IEEE Access*, 2020, 8: 46373-46399.

[30] Xiong X , Zheng K , Lei L , et al. Resource Allocation Based on Deep Reinforcement Learning in IoT Edge Computing[J]. *IEEE Journal on Selected Areas in Communications*, 2020, 38(6): 1133-1146.

[31] Wu J H , Li G S , Lin Q Y , et al. Resource Management Framework Based on the Stackelberg Game in Vehicular Edge Computing[J]. *Complexity*, 2020,vol. 2020: 1-11.

[32] Guo S , Dai Y , Guo S , et al. Blockchain Meets Edge Computing: Stackelberg Game and Double Auction Based Task Offloading for Mobile Blockchain[J]. *IEEE Transactions on Vehicular Technology*, 2020, 69(5): 5549-5561.

[33] Montes J M , Ramirez C E , Gutierrez M C , et al. Smart Contracts for supply chain applicable to Smart Cities daily operations[C]. *Casablanca: ISC2*, 2019: 565-570.

[34] Ni Yuandong, Zhang Chao, Yin Tingting. A Survey of Smart Contract Vulnerability Research[J].*Journal of Cyber Security*, 2020, 5(3): 78-99.
(倪远东, 张超, 殷婷婷.智能合约安全漏洞研究综述[J].*信息安全学报*, 2020, 5(3):78-99.)

[35] Li Zichen, Zhang Juanmei, Yang Yatao, et al. A Fully Homomorphic Encryption Scheme Based on NTRU[J]. *Acta Electronica Sinica*, 2018, 46(4): 938-944.
(李子臣, 张卷美, 杨亚涛,等. 基于NTRU 的全同态加密方案[J]. *电子学报*, 2018, 46(4): 938-944.)

[36] Yang Yatao, Zhao Yang, Zhang Qilin, et al. Weighted Electronic Voting System with Homomorphic Encryption Based on SEAL[J]. *Chinese Journal of Computers*, 2020, 43(4): 711-723.
(杨亚涛, 赵阳, 张奇林, 等. 基于 SEAL 库的同态加权电子投票系统[J]. *计算机学报*, 2020, 43(4): 711-723.)

杨亚涛 (1978–), 河南人, 博士, 教授, 硕导。主要研究领域为密码学与通信安全、同态加密、密码协议和算法等。E-mail: yy2008@163.com

林天祥 (1996–), 浙江台州人, 北京电子科技学院硕士研究生。主要研究领域为区块链安全、安全协议与算法。

陈剑源 (1997–), 山东菏泽人, 北京电子科技学院硕士研究生。主要研究领域为信息安全、密码协议与算法。

曾萍 (1969–), 河南人, 博士, 教授, 硕导。主要研究领域为通信与网络安全,车联网安全, 区块链安全等。

刘培鹤 (1972–), 黑龙江鹤岗人, 工程师。主要研究领域为网络与通信安全、区块链安全。