计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(7):12−23 [doi: 10.15888/j.cnki.csa.007457]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于区块链的数据管理设计模式①

姚浩男, 卢清华, 张卫山, 刘　越

(中国石油大学 (华东) 计算机科学与技术学院, 青岛 266580)
通讯作者: 卢清华, E-mail: dr.qinghua.lu@gmail.com

摘　要: 区块链作为一种新兴的技术, 由于其去中心化, 透明性和不可篡改性而受到广泛关注. 对区块链的研究还处于早期阶段, 应用程序开发过程中仍然存在大量的数据管理问题, 如数据隐私、可伸缩性和延迟. 在本文中, 我们提出了区块链数据管理的设计模式, 即数据隐私保护模式, 哈希完整性模式, 状态通道模式, 帮助开发者易用区块链进行应用开发. 上述所有模式都在一个基于区块链的溯源系统——originChain 中实现并验证, 并对其效果进行了讨论.

关键词: 区块链; 数据管理; 智能合约; 共享架构; 溯源系统

引用格式: 姚浩男,卢清华,张卫山,刘越.基于区块链的数据管理设计模式.计算机系统应用,2020,29(7):12–23. http://www.c-s-a.org.cn/1003-3254/7457.html

Design Patterns for Data Management of Blockchain-Based Systems

YAO Hao-Nan, LU Qing-Hua, ZHANG Wei-Shan, LIU Yue

(College of Computer Science and Technology, China University of Petroleum, Qingdao 266580, China)

Abstract: As an emerging technology, Blockchain has gained much attention due to its decentralization, transparency, and non-tamperability. Because the study to Blockchain is still in its early stages, there are still a number of data management issues in the application development process, such as data privacy, scalability, and latency. In this study, we propose the design patterns of Blockchain data management, namely, data privacy protection pattern, hash integrity pattern, and state channel pattern, to help developers use Blockchain for application development. All the above patterns are implemented and verified in a Blockchain-based traceability system: originChain, to show the effectiveness of the proposed patterns.

Key words: Blockchain; data management; smart contract; shared architecture; traceability system

比特币 (Bitcoin)[1]是一种基于对等网络和加密技术的数字货币. 区块链是比特币背后的技术, 提供不可变的和共享的数据存储, 并且事务数据是防篡改的, 与传统的数据库系统[2]相比, 它允许插入和查询事务, 阻止更新或删除事务操作. 区块链网络中的所有节点都可以在交易数据的状态上达成协议, 而无需依赖于一个集中化的系统.

数据管理是有效地收集, 存储, 处理和应用数据的过程. 随着数据存储冗余的不断减少, 数据独立性的不断增强以及数据操作的更加方便和简单, 数据管理得到了长足的发展. 区块链技术的出现为数据管理提供了一种新的方式, 在安全, 数据隐私和身份认证等方面都得到了广泛的研究.

然而, 由于区块链研究还处于早期阶段, 仍然存在许多限制. 例如, 大数据集很难用块存储; 区块链数据分布共享, 可能导致信息泄漏; 而且, 如果没有良好的数据管理模式设计, 处理大量需要共识的事务通常会很慢.

① 收稿时间: 2019-11-21; 修改时间: 2019-12-16; 采用时间: 2019-12-20; csa 在线出版时间: 2020-07-03

因此, 本文提出了改进区块链数据管理的几种设计模式, 包括哈希完整性模式, 数据隐私保护模式和状态通道模式. 设计模式在很大程度上简化了开发过程, 提高了区块链应用程序开发人员的效率. 我们实现了一个真实的基于区块链[3]的可跟踪系统 originChain[4], 作为一个案例研究来展示如何在实践中应用设计模式. 本文讨论了链上组件和链外组件之间的交互限制, 并就如何使用这些设计模式分享了一些经验.

本文的其余部分组织如下, 第 1 节介绍了基于区块链的应用背景和相关工作以及设计模式. 第 2 节讨论设计模式. 第 3 节讨论设计模式的应用. 第 4 节总结了本文的工作和未来的工作.

1 背景与相关工作

1.1 区块链技术

区块链作为比特币[1]的底层技术, 是一种用于存储交易记录的去中心化, 公开透明的分布式数据库[5]. 它是一种融合了分布式数据存储, 点对点传输, 共识机制和密码学[6–9]的新技术. 区块链最初只提供一个公共分类账来记录与特定数字加密货币相关的交易, 其工作领域仅限于金融业. 由于引入了名为智能合约[10]的可编程基础设施, 区块链能够处理复杂的事务处理, 比如触发器、条件和支持应用程序的业务逻辑[11], 使其可以应用到更多的领域行业, 比如医疗[12,13]、物联网(IoT)[14,15]以及司法事务等.

区块链将数据存储在有序的块列表中. 区块链上的每个块都包含一个事务列表[16], 区块之间链接通过后一个区块中记录前一个区块哈希值的方式进行. 且在区块链上已发布的事务不能被删除或更改, 由于哈希值的唯一性, 如果一个块中的交易被更改, 那么该块的哈希值会相应的发生变化, 导致后续已生成的所有块的哈希值都会相应发生变化, 这是一个巨大的工作量, 区块链网络中的相邻对等节点也会检测此变化, 并阻止此变化且更正此修改. 区块链的这种特性使其具有不可篡改性.

区块链网络中的节点是相等的, 区块链通过共识算法[17]解决节点之间的信任问题, 体现了区块链的去中心化特性. 区块链的去中心化可以在实际应用中解决信任问题, 例如通过不可信的第三方支付问题, 点对点区块链基础结构避免了第三方的信任管理.

智能合约: 智能合约最初是由 Nick Szabo 在 1994 年提出的. 智能合约是可编程的触发器、条件和业务逻辑, 以支持复杂的业务流程[18,19]. Solidity[20]是一种实用的智能合约编程语言. 从理论上讲, 智能合约可以对所有的业务流程进行编程, 这使得从区块链中开发应用程序成为可能. 智能合约是纯编程函数, 不能直接访问外部系统的状态.

以太坊: 以太坊[21]是一个开源的公共区块链平台, 可以部署智能合约, 为用户提供各种模块构建应用程序. 以太坊中的应用程序实际上是一个智能合约[22]. 用户通过智能合约编写的业务是不可预测的, 这为用户在以太坊上构建各种应用程序提供了灵活性和强大的功能, 但同时也带来了整个系统的不确定性和安全性以及性能问题. 因此, 设计模式对于区块链数据管理非常重要.

1.2 设计模式

设计模式[23]是可重用的微架构[19], 它们为在特定上下文中设计重复出现的问题提供了可靠的解决方案. 第一个也是最著名的一组设计模式是 Gamma 等人在书中提出的. 设计模式有 3 种类型, 包括创造性模式、结构模式和行为模式. 创造性模式涉及对象创建, 结构模式捕获类或对象组合, 行为模式处理类和对象分配职责和交互的方式.

设计模式支持代码的重用, 使代码更容易理解, 并确保代码的可靠性. 设计模式是软件开发人员在应用程序开发过程中所面临的常见问题的解决方案. 正确使用设计模式可以避免许多问题.

1.3 区块链数据管理

随着区块链的不断发展, 它对学术界和产业界都是一个机遇和挑战. 如何充分利用区块链的特点服务于行业成为各行各业的难点. 数据管理与各个行业紧密相关, 可以作为结合各个行业和区块链的路径.

在学术界, 学者们对区块链的特性以及如何利用这些特性使区块链[24]适应各行各业进行了相关的研究工作. Bartoletti 和 Pompianu[25]讨论了智能合约在每个区块链平台上的解释和应用, 基于比特币和以太坊的应用领域, 量化了智能合约的使用, 并分析了以太坊最常见的编程模式. Xu 和 Pautasso[26]提出了一种基于区块链的模式语言. 有些模式是基于区块链实际应用程序专门设计的, 而其他模式则是在智能合约上下文中

应用的现有设计模式的变体. 他们为区块链服务提出了 8 种设计模式. Eberhardt 和 Tai[27]提出了 4 种模式, 包括挑战响应模式、off-chain 签名模式、内容寻址存储模式、委托计算模式和低合同足迹模式, 这些模式主要关注于数据和计算的链上和链下分离. 然而, 上述相关研究只是通过理论分析来分析区块链在数据管理中的可行性. 在本文中, 我们重点介绍了设计模式的实现, 并给出了在实际应用程序中的应用. 代码已上传至开源网站 Github[28].

2 数据管理设计模式

在本节中, 我们为基于区块链的应用程序的性能, 隐私和成本设计并实现了以下 3 种模式. (1) 哈希完整性模式确保任何大数据集的完整性和不可变性, 大文件不直接存储在区块链中. (2) 数据隐私保护模式对区块链应用程序中的机密数据进行加密, 将密文存储在链上, 只有具有权限的人才能进行访问. (3) 状态通道模式, 允许执行链下事务, 将小额支付的高成本支付行为转移到链下, 只在链上执行状态更新. 数据管理设计模式结构图如图 1 所示.

如图 1 所示, 哈希完整性模式存储哈希值可以通过智能合约方式进行存储, 也可以通过小额支付状态通道模式的方式, 通过将哈希值附属在转账交易的信息中, 直接存储到区块链账本中. 两种方式最终的验证方式不同, 智能合约方式可以通过针对性的键值获取哈希值, 小额支付状态通道模式是通过交易哈希值获取交易信息, 然后在获取存储的哈希值. 状态通道模式致力于小额支付的应用, 也适用于奖励机制的创造, 如 3.4 节中所描述的场景. 隐私保护模式是将数据以密文的形式存储到区块链上, 密文可以是公钥加密过后的密文, 也可以是数据的哈希值 (通过使用哈希完整性模式计算出的哈希值), 这两种方式都可以保证数据的隐私不被泄露, 通过链上智能合约与链下数据库中的数据表一一对应的方式存储, 方便用户的存储以及查询. 下文是对 3 种设计模式的详细描述.

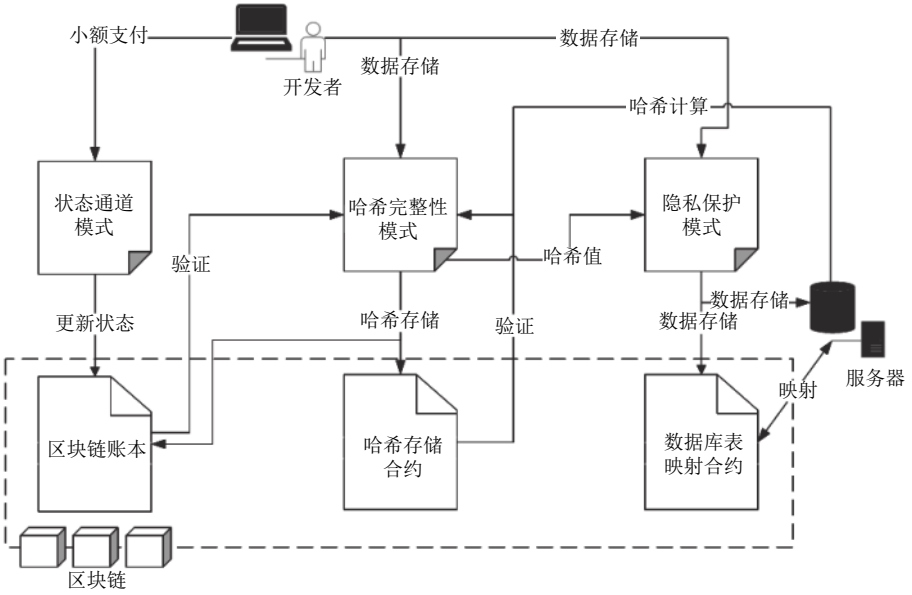

图 1 数据管理设计模式结构图

2.1 哈希完整性模式

摘要: 不直接将大数据集或文件存储在区块链上, 而是将数据集或文件的散列放在链上, 以确保数据的完整性.

场景: 区块链的特点和作为存储数据库的能力促使一些应用程序使用区块链来保证数据集的不变性和完整性. 任何数据集都可以存储在区块链中.

问题: 在区块链上存储数据时, 存储的数据越大, 所需的 gas (以太坊交易指标) 值越大, 所需的时间也越多, 相应的也会增加整个区块链的负担, 使得整个区块链的存储量增加巨大. 对于某些大数据集或文件, 即使给出了较大的 gas 值, 也可能由于块大小而无法存储

在区块链中. 区块链网络中的所有节点用户都将保留完整的数据块, 由于区块链的存储能力, 这将成为用户的负担. 区块链中的块大小是有限的, 并且可能无法在一次交易事务中存储大型数据集或文件. 如何在区块链上存储任意大小的数据, 以保证数据的不变性和完整性, 是区块链数据管理过程中的一个重要问题.

解决方案: 哈希函数有一个特性: 由同一哈希函数产生的两个不同的哈希值, 其所指示的原始数据也是不相同的. 通过哈希函数的特性和区块链的特点, 可以帮助解决数据存储在区块链上时的数据量大的问题, 也可以保证数据的不变性和完整性.

在进行大数据集存储的时候, 将原始数据存储在链下的数据库, 然后将数据的哈希值存储到区块链上. 以存储文件为例: 用户获取文件, 然后对文件进行哈希运算操作, 将文件以及文件的哈希值存入到链下数据库中, 然后以文件名作为键, 文件的哈希值作为键值, 存储到区块链上.

对于存储到链上的哈希值如何使用, 在模式中提供了验证的功能. 验证文件是否被修改过, 需要将文件再次哈希, 然后根据文件名获取到链上存储的哈希值, 两者进行比对, 使用的是同一哈希函数进行的文件哈希, 所以如果文件进行过改动那么两次的哈希值是不同的, 然后将结果反馈给用户. 哈希值的存储以及哈希值的获取比较, 充分利用了区块链的不可篡改性和哈希的不变性, 解决了大数据无法存储到链上的问题, 保证了数据的不变性和完整性.

结构: 哈希完整性模式的结构如图 2 所示.

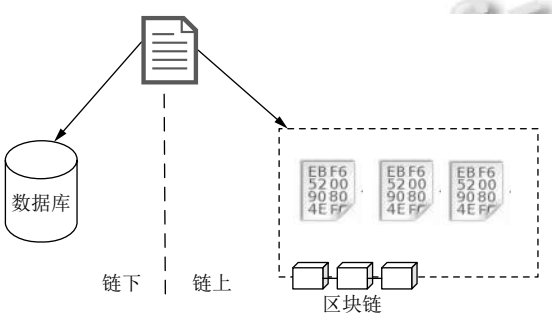

图 2 哈希完整性模式结构图

结论:

(1) 优点: 区块链保证了原始数据的完整性. 通过区块链上已存储的哈希值可以检查原始数据的完整性.相对于将大数据集或文件存储在区块链上, 例如某些应用开发者, 将整个数据库的压缩文件存储到区块链上; 司法公正相关文件, 包括图片, 证物文件等, 证据文件完整的存储到区块链. 以上常见方式会造成的区块链数据量的增加负担, 用户需承担的巨额 gas 消耗, 而且可能由于单个文件过大, 超过单个区块的存储量, 导致文件无法存储的情况发生. 然而只存储固定长度 (256 位) 的一串字符串 (哈希值) 到区块链, 很大程度的减轻了区块链存储负担及 gas 花费负担, 由于存储在链上的是数据哈希值, 所以可以保证任意数据大小的完整性, 不在受区块链块大小的影响.

(2) 缺点: 原始数据是链下存储, 无法保证原始数据的安全性以及不被修改. 只能通过比对哈希值的方式检测数据是否被修改, 如果数据一旦被修改则无法进行恢复.

(3) 注意事项: 需要专门的一个模块来计算数据的哈希值.

已知应用:

存在证明 (POEX. IO)[29]是一种用于私有区块链上使用 MultiChain 和 AWS EC2 的文件的加密工具. 证明存在提供的服务允许在比特币区块链中输入 SHA-256 加密的文档散列, 作为文档存在的证明, 同时保证数据的完整性.

Chainy[30]是一个运行在 Ethereum 区块链上的智能合约. Chainy 将一个到链下文件的短链接及其对应的散列值存储在一个位置. 它可以与存在证明相结合. 出于安全或隐私原因, 可能不希望信任将文件保存在属于第三方的服务器上. Chainy 通过将证明和文件链接保存在一个智能合约中解决了这个问题.

2.2 数据隐私保护模式

摘要: 区块链的一个特性是透明性, 因此它不能用于共享一些商业关键数据. 对于特定的数据使用密文的形式存储于区块链上, 对于不同程度或者不同方式的数据可选择对称加密或者非对称加密.

场景: 对于基于区块链的应用程序, 可能存在一些关键性数据 (例如银行金融方面的数据), 只有相关人员才可以有访问的权限. 例如审计部门对于溯源过程的审计结果, 只有颁发证书的相关部门负责人才可以访问到这个结果, 无关人员无法进行访问.

问题: 区块链的公开透明性是区块链的特点, 但这也使得重要数据的隐私问题受到挑战. 数据隐私问题是区块链的弊端, 区块链上的数据透明性导致存储在

区块链上的数据信息, 对于加入到区块链中的所有节点用户都是可以访问的. 区块链上的信息是公开透明的, 即使是新加入的节点, 也拥有一个区块链上的完整账本, 以及访问区块链上的所有信息数据的能力. 区块链中的节点都是平等的, 没有拥有特殊职权的节点. 区块链的数据透明性导致一些关键数据存储在区块链上时, 会发生数据泄露问题. 针对这一问题, 设计实现了数据隐私保护模式, 解决了用户数据隐私发生泄露问题.

解决方法:

为了保护用户的隐私, 在往链上存储数据的时候, 对一些关键的数据进行加密处理, 然后再存储到链上. 这里使用非对称加密的算法进行对数据加解密操作. 每个节点用户拥有一对公私钥, 将私钥进行自己保管, 公钥存储到链上公示. 如果自己存储数据, 可以用自己的公钥进行加密操作, 然后再将加密后的密文存储到链上; 如果是发送数据给别人, 可以从链上获取接收方的公钥, 然后用其公钥进行数据加密操作, 接收方再用自己的私钥进行解密查看. 这样就解决了关键数据在链上时, 会造成数据泄露问题. 虽然其他节点也会看到, 但他们看到的只是加密后的密文, 没有私钥是无法进行查看原数据的.

在进行数据库表创建时, 用户需要筛选出哪些属性是关键数据, 需要存储到链上, 与此同时会在链上创建与表相对应的智能合约, 用于存储这些关键数据. 链上的存储的键与链下数据库中的主键相对应, 由于存储以密文形式存储, 链上数据存储统一数据类型 (string 型), 对于数据类型之间的转换, 在链下进行, 会根据其在数据库中的存储类型进行对应的转换.

在进行数据存储时, 链下数据库中存储完整的表结构数据, 链上仅存储那些关键数据 (被筛选出需要存储在链上的数据). 根据用户存储数据的用途来选择是用自己的密钥加密, 还是用他人的密钥进行加密. 如果是自己作为存储使用, 那么获取自己的公钥进行数据加密操作, 然后将数据分别存储到链上和链下; 如果是某些数据需要发送给他人, 这时需要获取他人在区块链上公示的公钥进行加密操作, 然后进行存储到链上, 接收方只需要得到这个数据, 然后使用自己的私钥进行解密查看.

结构: 数据隐私保护模式的结构如图 3 所示.

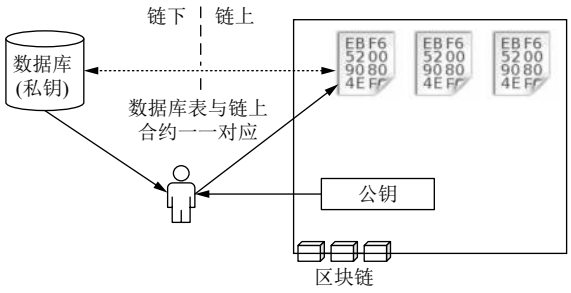

图 3 数据隐私保护模式结构图

结论:

(1) 优点: 保证了数据的隐私性. 存放在区块链上的关键性数据是经过加密的, 虽然区块链上的数据都是共享的公开透明的, 但其他节点如果没有秘钥是无法进行解密的.

(2) 缺点: 秘钥需用户自己存储, 用户如果导致秘钥泄露, 就会造成数据的隐私受到威胁. 加密过后的密文较长, 需要分段存储.

(3) 注意事项: 针对不同需要选择是用对称密钥还是非对称密钥; 密钥的产生需要另一个模块; 密钥的管理需谨慎, 防止泄露; 使用非对称密钥, 公钥需存储到区块链, 私钥用户自己存储, 不论使用对称或非对称密钥, 节点间共享加密的数据, 需要先共享密钥.

已知应用:

Oraclize[31]是一个运行在以太坊公网上的智能合约, 它提供了从外部世界访问状态的服务. 它在数据源中使用数据加密和解密操作. 开发人员可以使用 Oraclize 公钥加密整个查询或它的一些参数, 从而实现加密的 Oraclize 查询. 唯一能够解密调用参数的是使用配对的私钥 Oraclize.

MLGBlockchain[32]是一家全球性的风险创造和咨询公司, 拥有区块链技术开发和经纪经销商的能力. 由 MLGBlockchain 提出的加密数字签名, 用于加密数据, 并在通过区块链交互和传输数据的各方之间共享数据.

2.3 状态通道模式

摘要: 区块链上每一笔交易的发生都会产生资金资源消耗, 小型的交易 (包括小额的转账交易或者单个数据交换等) 不适合在区块链上进行, 将此类型的交易转移到链下, 等待一段时间或者一定交易次数达到来定期更新链上的状态.

场景: 小型交易包括小到几美分的资金交易例如

WiFi 热点的收费, 每使用 1 万字节的数据, 就会收取一小笔钱; 或者单个数据的交换发生, 例如一组测试样品的实验室之间的测试结果的交换. 问题是, 是否有必要将这类交易全部存储到区块链, 占用区块链的大量存储空间.

问题: 区块链上存储数据需要耗资, 个人小额支付交易的交易费用可能高于与小额支付交易相关的货币价值. 区块链上的交易需要确认, 而确认时间并不能确定, 但用户期望的是小额交易支付能够即时完成. 区块链上的交易时间长, 费用高, 以致不可能在区块链上存储每一笔小额的交易. 在这么多交易矛盾的情况下, 是否有必要在区块链上进行小额交易成为一个问题.

解决方法: 小额交易的成本太高, 在区块链上存储小额交易是不可行的. 但小额交易的发生是不可避免的, 为了解决小额交易的问题, 本文中设计实现了状态通道模式. 状态通道指小额交易的发生持续进行, 只在链上更新最终的交易状态. 交易的终止点会有限定, 当达到限定条件后终止交易, 更新链上状态.

状态通道有两种限制, 一是设定个人的交易限制额度, 一旦交易超过限制额度则停止交易关闭通道, 更新链上状态; 二是设定个人交易次数, 次数达到停止交易关闭通道, 更新链上状态. 状态通道初始是无通道的, 一旦交易发生, 先会查找是否有可以借用的他人建立的通道, 如果存在则借用别人通道进行交易的传输; 如果查找不到可借用通道, 则交易双方建立直接的一条交易通道. 满足可借用通道的条件是交易额度不受通道限制 (每一条通道会有一个“带宽”), 中间节点的余额满足交易额度. 每一条通道会有一个“通道带宽”, 限制通道允许交易的额度, 是由通道两端节点在第一次建立通道时随机产生的一个大于第一次交易额度的值. 交易进行前会对交易双方的账户余额进行查询验证看是否满足交易额度, 满足则通过状态通道进行交易, 否则无法进行交易.

链下的交易不受区块链上块大小、远近的限制, 也不受确认块的时间限制, 无须等到所有区块链节点达成共识才完成交易, 只在链上更新所有交易完成后的最终状态. 而小额交易的细节不公开, 不记录在链上账本中, 链上账本只记录最终的交易. 极大地节省了不必要耗资, 提高了效率.

结构: 状态通道模式的结构如图 4 所示.

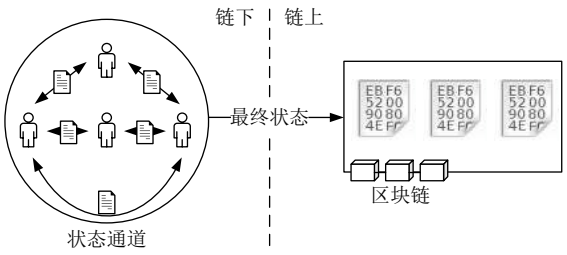

图 4 状态通道模式结构图

结论:

(1) 优点: 避免了过度的资源的消耗, 减少了交易所消耗的时间, 提高了交易的性能; 只在最终交易完成以后更新链上的状态, 交易细节不在公布; 由于在链下进行交易, 交易不在受限于区块链上块的大小的限制.

(2) 缺点: 通道交易额度的限制, 需根据具体情况进行调节, 每次交易的发生可能会不断更新通道限额, 影响交易的效率.

(3) 注意事项: 链下需要一个类似于以太坊钱包的控件, 用来更新存储链上的状态.

已知应用:

Lightning Network[33]是一种链式服务解决方案, 它不发送任何货币, 而是在第 2 级更改分类帐, 然后在第 1 级完成结算, 从而避免了数千次实际的资金转移. 闪电网络通过将资金发送到由多方托管的多个签名地址来构建支付渠道. 付款通道可以通过播放最终版本的融资交易来关闭, 以结算付款.

3 在溯源系统中的应用

在本节中, 以溯源系统[4]为应用实例, 来验证在第 2 节中提出的 3 个设计模式, 并对其进行详细的使用描述.

3.1 溯源系统

溯源系统[4]指物联网、移动互联网以及一物一码等技术的整合应用. 在产品生产过程中采集产品数据信息并形成溯源档案. 在产品生产和分销中跟踪产品相关信息 (原料产地、生产加工、质量检测以及物流运输) 是溯源系统的主要目标. 由政府部门认证的溯源公司, 可以向原料提供方、产品生产商以及零售商提供溯源服务, 溯源公司派遣相关人员, 到产品的各个环节进行考核检查, 并且对于各个环节的样品进行样品检测, 如果符合政府部门发布的合格标准, 则予以颁发溯源证书, 否则将拒绝颁发. 图 5 演示了产品可追溯性的业务流程[19].

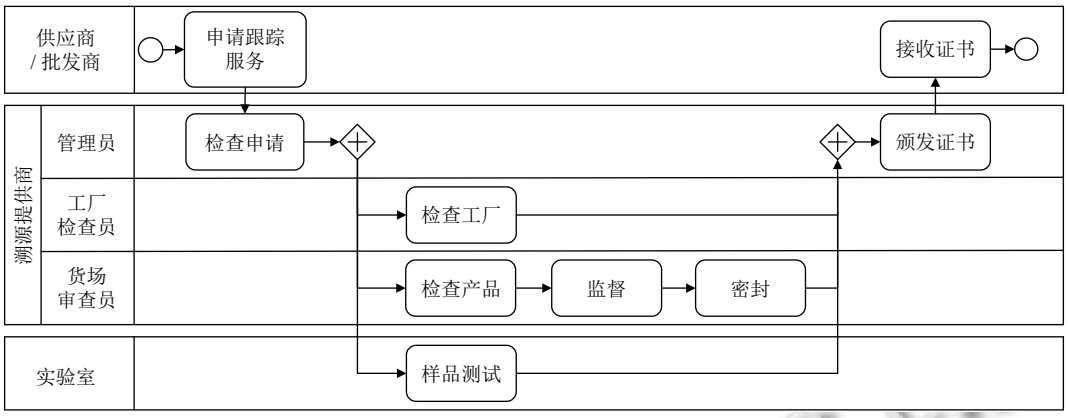

图 5 溯源服务过程图

基于区块链的产品溯源系统 (originChain)[4]是由传统数据库与区块链技术结合形成. 供应链行业[34]需要数据的透明性以及不可篡改性, 以确保产品信息的真实性和可靠性. 正因为区块链技术的特点, 使得区块链技术适用于溯源系统的各个环节.

3.2 哈希完整性模式应用

溯源证书是可以被修改或者被恶意替换的, 存在链下数据库中总是不安全的, 如何保证溯源证书在传输过程中的完整性和不变性是个问题. 随着区块链的问世, 区块链的特点为解决此类问题提供了方案.

溯源公司在确定商品后, 给相关部门颁发溯源证书[15], 并对溯源证书进行哈希计算, 将哈希值存储到链上. 然后供货商接收到溯源证书, 将溯源证书贴于商品上; 购货商购买商品, 如果想验证真伪, 需将溯源证书再次哈希, 然后获取存取在链上的哈希值进行比对, 如果结果一样说明溯源证书未被修改过, 否则溯源证书被修改过. 以此来检验溯源证书的真实性.

相关实现类如图 6 所示, DB_Test 类包含 insert 和 select 方法, 为用户链下数据和文件的存储与查询提供接口; DBConnection 类是数据库的连接类, DB_Test 类中的方法通过调用 DBConnection 中的连接对象进行数据库操作; PdfreadServlet 类用于用户页面交互, 对于页面上用户相关操作通过该类进行实现; Pdfdpo 类用于与数据库中的数据表相对应的实体类; MyPdfReader 类, 用户对数据文件进行 Hash 操作的实现类; CompareFileServlet 类, 用于 Hash 值的比对实现类, 用于页面交互, 返回页面数据.

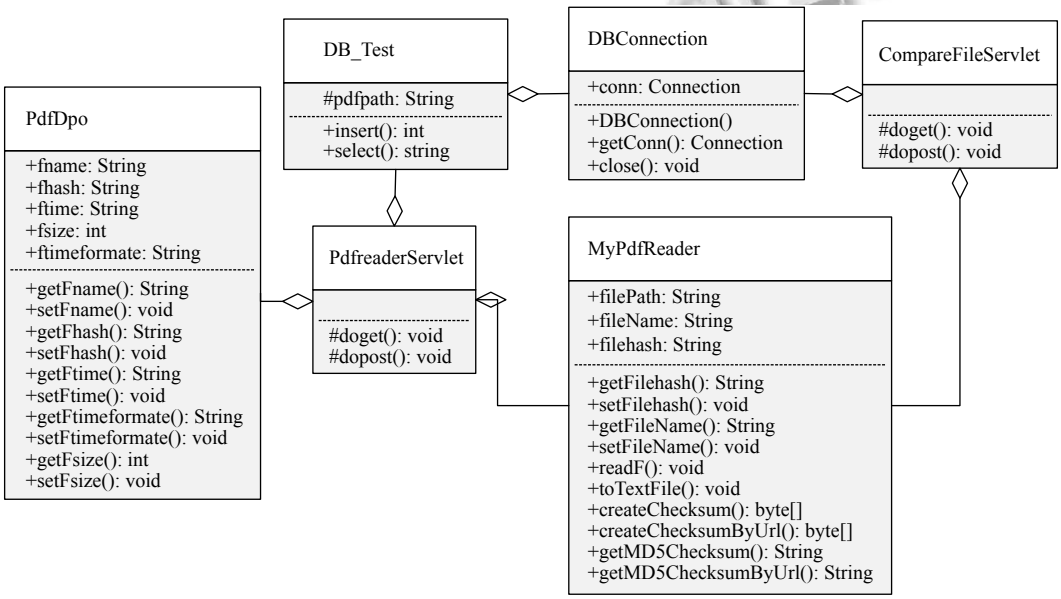

图 6 哈希完整性模式类图

具体实现操作以溯源证书的颁发为例展示类与类之间的调用. 溯源公司将溯源证书通过 PdfreaderServlet 类进行 Hash 操作, PdfreaderServlet 调用 MyPdfReader 类中的 getMD5Checksum 方法对溯源证书进行 Hash 计算, 然后 PdfreaderServlet 将得到的 Hash 值存储到数据库中, 再将 Hash 值传递回页面, 然后将 Hash 值存储到区块链上, 完成文件数据 Hash 存储操作. 购货商购买商品检验真伪, 可以将商品上对应的溯源证书通过 CompareFileServlet 类进行比较在确定真伪, 购货商将溯源证书通过 CompareFileServlet 通过调用 MyPdfReader 中 getMD5Checksum 方法进行 Hash 计算, 然后通过获取相应的链上已经存在的商品 Hash 值, 两者进行比对, 如果 Hash 值一样, 则返回用户真, 否则返回用户否. 这样就完成了对具体实例的验证操作.

3.3 数据隐私保护模式应用

在溯源过程中, 政府的审计部门会对相关产品的一系列类型以及溯源双方的货物定价会进行审核. 例如进口奶粉, 溯源服务公司跟奶粉供应商签订合同, 合同中包含溯源服务的类型和价格, 政府部门需要审计这个合同, 然后将结果提交到中国质检总局验证. 结果是私密的, 不能被其他人所看到, 链上数据存储对任何人都是透明的, 所以解决私密数据泄露是审计过程的难题. 数据隐私保护模式解决了这一问题.

审计过程中审计部门对溯源双方所签署的合同进行审计, 然后将审计结果使用政府部门的公钥 (从链上获得) 进行加密, 然后将密文存储到链上相关的合约中, 政府相关工作人员在验收的时候使用政府部门的私钥进行解密查看结果. 这样避免了在链上存储关键数据时, 数据的泄露问题. 数据的加密, 根据具体情况自行选择加密方式. 溯源公司的内部数据可以使用自己的公私钥进行加解密, 如果涉及到其他的问题, 比如审计部门审计, 审计结果只能让相关部门的工作人员查看到, 这种针对性的需要获取他人的公钥进行加密, 根据不同情境进行不同的变换, 充分的保证了数据隐私泄露问题.

相关实现类如图 7 所示. SelectDataServlet 类用于查询数据时候的页面交互; InsertDataServlet 类用于用户存储数据时候的页面交互; DBConnByname 类用于数据库连接操作; DecryptedataServlet 类用于对数据进行解密操作的交互; EncryptedataServlet 类用于对数据进行加密操作的交互; Encryption 类用于实现加解密的具体操作, 包括生成对称或者非对称密钥, 以及数据加密操作和解密的具体实现操作.

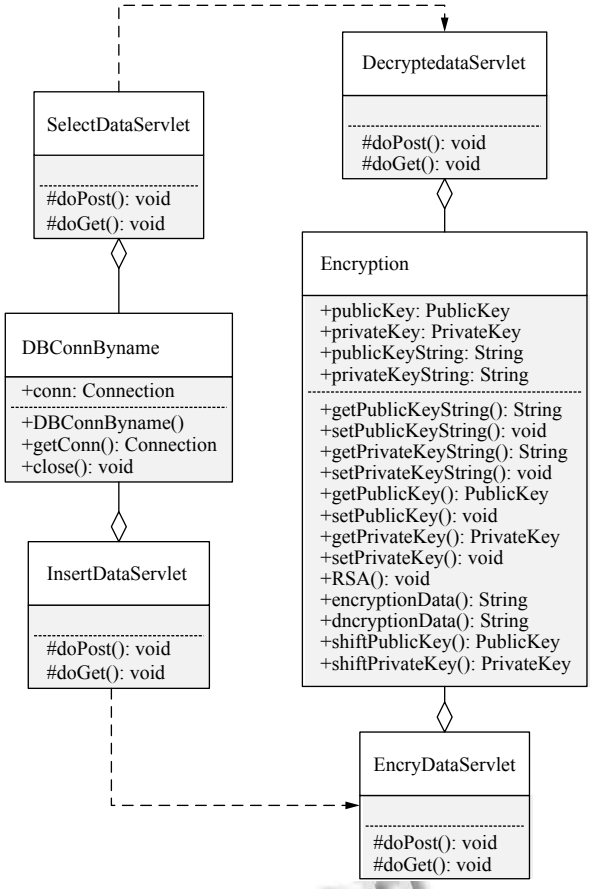

图 7 数据隐私保护模式类图

具体的实现操作以审计部门的审计工作作为实例, 描述类之间的交互. 审计部门人员将审计结果使用政府部门的密钥进行加密, 首先通过页面获取区块链上已经存在的政府部门的公钥, 将公钥和审计结果一起传递给 EncryptedataServlet, 然后调用 Encryption 中的 encryptionData 方法返回加密过后的审计结果, 然后 EncryptedataServlet 再将加密后的结果返回页面, 首先将结果存储到数据库中, 通过 InsertDataServlet 类进行存储, 然后再将结果存储到区块链上, 结束审计部门审计工作. 政府查看审计结果, 首先从区块链上获取审计部门的审计结果, 然后使用自己的私钥进行解密操作. 页面通过 SelectDataServlet 类从数据库中获取存储的私钥, 然后将私钥返回页面, 页面将获取的链上审计结果以及私钥传递给 DecryptedataServlet 类, DecryptedataServlet 类通过调用然后调用 Encryption 中的 decryptionData 方法返回解密过后的审计结果, 通过

DecryptedataServlet 类将明文返回页面, 显示给用户查看. 这样从审计到审计结果的验收工作就实现完成.

3.4 状态通道模式应用

在溯源系统中, 样品的测试检验是至关重要的一环, 溯源商品质量过不过关需要有检验, 测试检验的结果符不符合质量标准是溯源商品值不值得信赖的关键保证. 在样品测试过程中, 可能涉及多个实验室进行交叉测试, 然后再将最后的结果作为商品的测试结果. 每个实验室在进行测试的时候会有相应的奖励作为报酬. 每个实验室的测试结果以及报酬的分别的存储到链上, 会造成大量的资源浪费, 因为每个实验室不可能只检验一个样品, 对于不同的样品的结果分开存储, 或者每检验完一个样品就通过区块链奖励一次, 这种情况下造成大量的区块链资源的浪费. 通过状态通道, 实验室之间相当于链下的不同节点, 不同实验室之间进行不断的实验结果的交换使用, 最后将最终的检验结果完整地发布到链上. 对于每个实验室的检验奖励作为积累, 当检验结束, 将最终的奖励进行一次性转账交易的记录.

如图 8 所示, MoneyServlet 与 TimeServlet 是获取链上每个实验室的资金状态页面交互; TransServlet 与 TimeForTrans 是初始化状态通道与页面交互; GoTransServlet 是实现状态通道操作的交互类, 进行的是资金限制操作相关操作类; TimeGoServlet 是实现状态通道操作的交互类, 进行的是次数限制相关操作类; CloseTransaction 用于最终的更新链上状态交互类; AssistClass 用于找到一条可以使用的通道供交易使用的辅助类; FindPath 类用于查找是否有通道可以借用的辅助类; Graph 用于图网络的建立, 我们将状态通道与图结构进行结合.

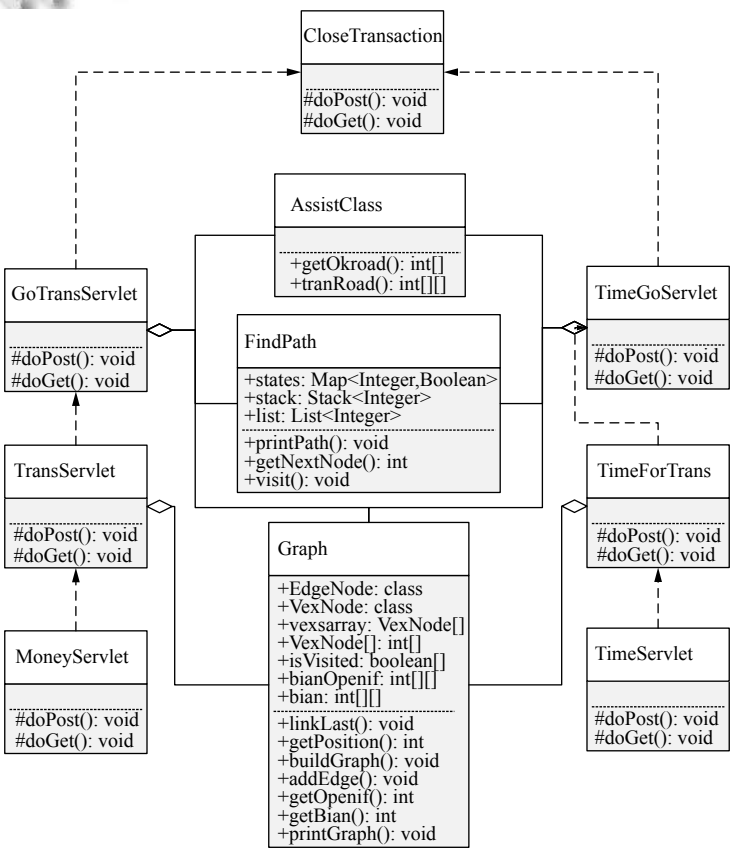

图 8 状态通道模式类图

具体实现操作以实验室的检测测试奖励为例来进行描述. 实验室测试样品, 如果完成一次测试, 实验室获得一次奖励, 那么当多个是现实多次发生类似情况的时候, 积累过多会给区块链造成沉重的负担. 将每一次的奖励在链下进行, 然后将最终测试结束后, 实验室的资金状态, 作为一次总的交易发生存储到链上. 首先

进行前先确定是资金方面的交易, 通过 MoneyServlet 进入到相应页面, 然后通过页面进入 TransServlet, 调用 Graph 类 buildGraph 进行状态通道的初始化, 初始化完成后就可以进行测试奖励的交易. 一次奖励的开始调用 GoTransServlet 进行操作, 首先调用查看是否有直接的通道, 如果有查看是否处于开启状态 (通道有可能处于关闭或者开启状态, 开启状态可以使用, 关闭状态则不能使用), 则直接进行奖励交易, 否则先开启通道在进行交易. 如果没有调用 FindPath 查找有没有其他的实验室节点通道可以借用, 如果有则调用 AssistClass 类, 查找是否有可用的通道, 只有处于开启状态的通道才可以进行借用. 如果找到则借用该通道进行交易, 如果没有找到则在奖励人节点与被奖励的实验室节点直接建立直接的通道并开启, 进行奖励的交易. 实验室测试奖励的交易以这种链下状态通道方式进行, 当所有测试完成, 则调用 CloseTransaction 进行状态的确定, 然后将最终的奖励交易作为一次总额度的交易进行产生, 更新到链上, 完成此过程. 如果是测试结果的交互使用, 则如同奖励实验室测试过程一样, 调用另一个分支即可完成.

3.5 应用设计模式开发效率分析

在上述部分中, 我们着重描述了将我们的设计模式应用到具体案例中的实现操作, 在本节中, 我们将对如何使用提出的设计模式进行应用程序开发, 及如何提高开发效率进行分析.

在使用本文所提出的设计模式进行区块链应用程序开发时, 开发人员只需运用本文提出的设计模式, 而无需担心其他区块链相关操作. 在哈希完整性模式中, 提供了完整性的哈希计算方法, 以及链上智能合约完整代码. 应用开发者只需要使用提供的哈希计算方法进行数据集的哈希计算, 然后将智能合约代码部署到本地区块链上, 即可存取数据. 在数据隐私保护模式中, 提供了完整的公私钥创建方法, 以及公私钥与字符串之间转换方法, 以便应用开发者可以方便地存储到数据库以及区块链上, 无需担心公私钥与字符串之间转换问题, 相应的我们还提供了存储公钥的智能合约, 以及存储方法. 我们还提供了链下数据表与链上智能合约一一对应相关操作, 应用开发者只需要在创建链下数据表的时候, 将需要存储到区块链中的属性筛选出来, 就会相应的生成与链下数据表名所对应的智能合约 (合约名与表名一致), 以及应用开发者所筛选出来的相关链上属性, 以及先关 get 和 set 方法. 应用开发者只需要使用相关模块即可进行应用的快速开发. 在状态通道模式中, 提供了预设条件值, 当交易发生 8 次或者间隔 8 秒钟之后, 就会自动的更新链上状态, 应用开发人员只需要设置自己预设的次数或者时间即可, 这里建议时间设置小一点, 减小被攻击的可能性. 状态通道模式一般应用于带有奖励机制的应用开发过程中, 具体的扩展性我们将进行进一步的研究.

设计模式中提供所有的方法以及智能合约模板, 尽可能简化区块链应用开发者的操作. 区块链应用程序开发者不再需要专门去学习如何书写智能合约, 只需要在提供的基础模板 (有相应解释注释) 上面做微小调整即可使用, 而且在提供的方法中, 大部分都是智能自动化生成的智能合约, 应用开发者只需要调用相应的方法即可, 节省了应用开发者再学习的时间, 提高了应用程序开发效率. 在应用开发过程中开发者也不需要专门的研究如何调用区块链的 API, 对于连接区块链操作, 发布区块链交易操作, 智能合约的部署, 访问查询操作等都已经进行了封装, 开发者可以通过使用方法的过程中了解, 其中代码都加注了相关接口注释. 我们致力于打造一个方便高效的模式, 使得一个不了解区块链的应用开发者也可以进行区块链应用程序开发, 通过节省开发者再学习的时间, 来提高应用程序开发效率.

4 总结

在本文中, 我们提出并设计实现了几种基于区块链应用的数据管理设计模式, 并分享了将所提出的设计模式应用于一个名为 originChain 的基于块链的可追溯系统的经验. 区块链的独特属性, 为数据管理提供了新的思路. 而设计模式影响基于区块链的应用程序的某些特定方面, 如可更新性、适应性和互操作性. 并且设计模式为开发人员在区块链上构建应用程序提供了架构指导. 我们根据区块链在数据管理方面的问题, 将设计模式分为了 3 类: 哈希完整性模式、数据隐私保护模式和状态通道模式. 并给出了模式与模式之间的相关性描述.

在未来的工作中, 我们将继续完善和扩展设计模式, 并提供有关设计模式更加详细的讨论. 将这些设计模式应用于更多适用区块链的领域. 我们将这些模式封装成了服务, 以供所需用户使用, 提供 API 以方便用户访问.

参考文献

1 Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008. Consulted, 2008. https://bitcoin.org/en/bitcoin-paper.

2 Han J, Haihong E, Le G, *et al*. Survey on NoSQL database. Proceedings of the 6th International Conference on Pervasive Computing and Applications. Port Elizabeth, South Africa. 2011. 363–366.

3 Weber I, Lu QH, Tran AB, *et al*. A platform architecture for multi-tenant blockchain-based systems. Proceedings of 2019 IEEE International Conference on Software Architecture (ICSA). Hamburg, Germany. 2019. 101–110.

4 Lu QH, Xu XW. Adaptable blockchain-based systems: A case study for product traceability. IEEE Software, 2017, 34(6): 21–27. [doi: 10.1109/MS.2017.4121227]

5 Muzammal M, Qu Q, Nasrulin B. Renovating blockchain with distributed databases: An open source system. Future Generation Computer Systems, 2019, 90: 105–117. [doi: 10.1016/j.future.2018.07.042]

6 Nasrulin B, Muzammal M, Qu Q. Chainmob: Mobility analytics on blockchain. Proceedings of the 19th IEEE International Conference on Mobile Data Management (MDM). Aalborg, Denmark. 2018. 292–293.

7 Nasrulin B, Muzammal M, Qu Q. A robust spatio-temporal verification protocol for blockchain. Proceedings of the 19th International Conference on Web Information Systems Engineering. Dubai, United Arab Emirates. 2018. 52–67.

8 Nurgaliev I, Muzammal M, Qu Q. Enabling blockchain for efficient spatio-temporal query processing. Proceedings of the 19th International Conference on Web Information Systems Engineering. Dubai, United Arab Emirates. 2018. 36–51.

9 Qu Q, Nurgaliev I, Muzammal M, *et al*. On spatio-temporal blockchain query processing. Future Generation Computer Systems, 2019, 98: 208–218. [doi: 10.1016/j.future.2019.03.038]

10 Omohundro S. Cryptocurrencies, smart contracts, and artificial intelligence. AI Matters, 2014, 1(2): 19–21. [doi: 10.1145/2685328.2685334]

11 Weber I, Xu XW, Riveret R, *et al*. Untrusted business process monitoring and execution using blockchain. Proceedings of the 14th International Conference on Business Process Management. Rio de Janeiro, Brazil. 2016. 329–347.

12 Mettler M. Blockchain technology in healthcare: The revolution starts here. Proceedings of the IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom). Munich, Germany. 2016. 1–3.

13 Zhang P, White J, Schmidt DC, *et al*. Applying software patterns to address interoperability in blockchain-based healthcare apps. arXiv: 1706.03700, 2017.

14 Ali MS, Vecchio M, Pincheira M, *et al*. Applications of blockchains in the Internet of Things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 2019, 21(2): 1676–1717.

15 Lo SK, Liu Y, Chia SY, *et al*. Analysis of blockchain solutions for IoT: A systematic literature review. IEEE Access, 2019, 7: 58822–58835. [doi: 10.1109/ACCESS.2019.2914675]

16 Fay T, Paniscotti D. Systems and methods of blockchain transaction recordation: US, 20160292672. [2016-10-06].

17 Wang WB, Hoang DT, Hu PZ, *et al*. A survey on consensus mechanisms and mining strategy management in blockchain networks. arXiv: 1805.02707, 2018.

18 Van Der Aalst W. Process mining: Discovery, conformance and enhancement of business processes. Berlin: Springer, 2011.

19 Weske M. Business Process Management: Concepts, Languages, Architectures. Berlin, Heidelberg: Springer, 2012. 333–371.

20 Community H. https://solidity-cn.readthedocs.io/zh/develop/. [2017-12-20].

21 Buterin V. https://www.ethereum.org/, 2015. [2017-10-30]

22 Seijas PL, Thompson SJ, McAdams D. Scripting smart contracts for distributed ledger technology. IACR Cryptology ePrint Archive, Report 2016/1156, 2016.

23 Pree W, Sikora H. Design patterns for object-oriented software development. Proceedings of the 19th International Conference on Software Engineering. Boston, MA, USA. 1997. 663–664.

24 Lo SK, Xu XW, Chiam YK, *et al*. Evaluating suitability of applying blockchain. Proceedings of the 22nd International Conference on Engineering of Complex Computer Systems (ICECCS). Fukuoka, Japan. 2017. 158–161.

25 Bartoletti M, Pompianu L. An empirical analysis of smart contracts: Platforms, applications, and design patterns. Proceedings of 2017 International Conference on Financial Cryptography and Data Security. Sliema, Malta. 2017. 494–509.

26 Xu XW, Pautasso C, Zhu LM, *et al*. A pattern collection for blockchain-based applications. Proceedings of the 23rd European Conference on Pattern Languages of Programs.

Irsee, Germany. 2018. 3.
27 Eberhardt J, Tai S. On or off the blockchain? Insights on off-chaining computation and data. Proceedings of the 6th IFIP WG 2.14 European Conference on Service-Oriented and Cloud Computing. Oslo, Norway. 2017. 3–15.
28 Yao H. https://github.com/cainiaohaonan/. [2019-02-23].
29 PoEx Co. Ltd. https://poex.io/prove. [2019-03-20].
30 Everex. https://chainy.info/. [2019-03-20].
31 https://provable.xyz/. [2019-02-25].
32 Leiberman B, Mirynech D. Digital Assets: The era of tokenized securities. Special Feature: Cutting-Edge Innovation in the Cryptosphere, 2019, 4(1): SF17–SF24.
33 Poon J, Dryja T. https://lightning.network/lightning-network-paper.pdf. [2019-01-10].
34 Mentzer JT, DeWitt W, Keebler JS, *et al*. Defining supply chain management. Journal of Business Logistics, 2001, 22(2): 1–25. [doi: 10.1002/j.2158-1592.2001.tb00001.x]