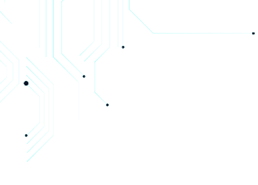
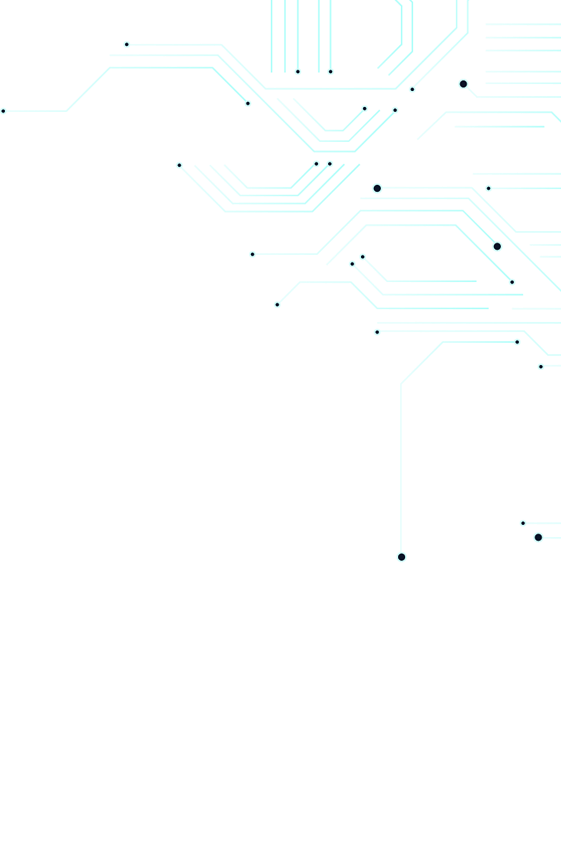
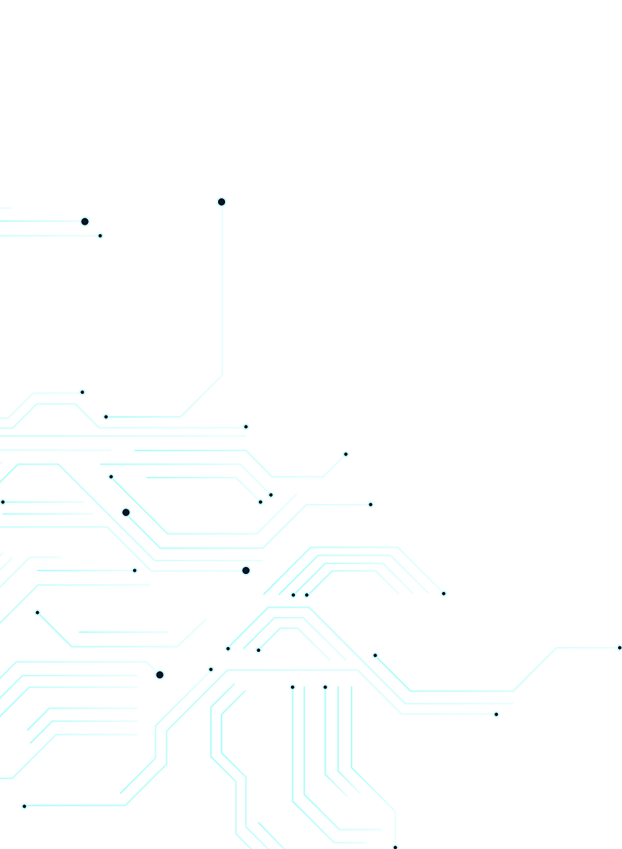
基于区块链的假新闻鉴定



FAKE NEWS

目录

**简单概述2**

研究背景2

现阶段方法的不足3

基于区块链的新闻鉴别的主要工作4

**相关技术的说明5**

区块链的介绍5

智能合约7

数字签名7

共识机制8

超级账本9

密匙协商协议10

**整体框架设计**11

新闻作者→审核员12

审核员→互联网13

互联网→用户14

结构导图15

**基于区块链的密钥及文档管理系统方案**16

密钥管理模块设计16

文档管理模块设计18

用户管理模块设计18

智能合约设计19

客户端设计24

**基于区块链的****文档管理系统实现与测试**29

系统开发环境及配置29

环境搭建29

二维码+区块链的实现34

系统功能测试36

**总结**37

# 简单概述

**研究背景**

随着信息科技的发展，新闻传播越来越容易，真新闻和假新闻鱼龙混杂。假新闻往往传播速度比真新闻更快，这是因为这些新闻看起来往往更具有说服力和冲击力，由于在互联网上传播假新闻无需任何成本和代价，这便给了假新闻传播者可趁之机。倘若对假新闻放任不管，很可能会导致各个领域个人和组织名誉受损的后果。假新闻，根据柯林斯词典的定义，指“假借新闻报道形式传播的错误虚假、耸人听闻的信息”。2017年11月2日，柯林斯英语词典2017年度热词揭晓——“假新闻”（fake news）加入科林斯词典的最新印刷版本。入选英国2018年度词语。

假新闻一旦传播到互联网上，就会留下相关的痕迹并持续发酵。想要尽力杜绝假新闻的发生，必须要从源头治理，由审核员对新闻作者严加审核，在假新闻传播之前就将其筛选掉，使得用户接收的新闻呢都是纯净的、真实的。

**现阶段方法的不足**

目前的新闻发布往往没有统一的审核渠道，既有政府的官方媒体，也有门槛极低的自媒体。正因为自媒体的门槛过低，大部分自媒体本着赚取流量的盈利目的，发布大量未经证实且粗制滥造的新闻。新闻公司和社交媒体正在努力缓解这一问题，其中有一些举措确实卓有成效，但这一问题仍在发酵——它已经超越了公司能够控制的界限。显而易见的是，有些机构会故意生产假新闻，以实现其政治、商业或其他目的。如前文所述，有一大群网络用户（可能是个人，也可能代表某一公司）会在无意中推动假新闻的指数传播。事实上，由于对问题缺乏了解，许多用户永远也不会发现他们已然置身于“假新闻系统”之中。想象一下每天都会运行的“内容抽样”过程，即对全球范围内发布和分享的内容进行抽样，这一过程得益于特殊的抓取工具，它可以在一组代表性网站、社交媒体和流行博客中“听取”各种故事和“新闻”，并将这些“新鲜内容”和“新内容索引”建构成一个统一的、不可复制且不可变的内容存储库，这一存储库专门用于处理各种故事、事实及其关联信息。新识别的内容被整合并关联至其“母本”、相关的“故事”及其他事实信息，然后与已标注内容进行比较，目的在于用“同一故事的真实版本”和已知模式来对该新闻“与现实的偏离程度”进行评估。随着人们网络意识的提高，越来越多的人开始思考新闻的真实性，现阶段的网络新闻管理普遍使用信息化管理模式，大部分网站采用的文档管理系统功能简单，通常只考虑内部文档的存储与管理，很少去验证新闻是否属实，随着新闻数据存储以及越来越多的假新闻的增加，旧的文档管理系统也有了分布式的需求。分布式文档管理系统在访问控制上存在一些不足,包括如何对接入系统的用户进行身份识别,如何存储与读取访问控制相关的数据。同时，旧的媒体没有深挖新闻背后的产业链，仅仅停留在，采访、写作、发布的阶段。

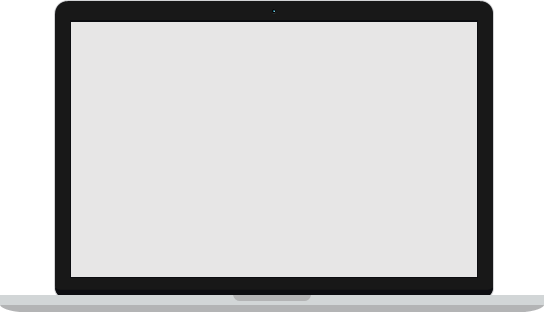
**基于区块链的新闻鉴别的主要工作**

结合分布式文档管理系统的特点，针对组密钥管理中计算量大、组密钥更新不同步、受限于管理节点可用性等不足,提出了基于区块链的密钥管理方案。方案采用密钥树的结构管理系统中的密钥,对用户而言，分层的树状结构便于管理用户组别与权限,密钥直接根据用户ID生成,简化了密钥生成与管理的复杂性。提出基于分层确定性钱包系统的组密钥更新方案,可以更高效的让新闻的作者和用户得到一致的组密钥,并且密钥更新过程不受管理节点的可用性影响，便于作者实时跟进与更新新闻。针对文档管理系统的特点，组密钥的更新是可以记录的,通过组密钥和存储在区块链上的组密钥更新参数可以计算得到所有的历史组密钥，用户可以以此来查看新闻的修改记录。本文提出的方案有分层分组式管理、节点存储密钥数少、密钥更新计算量小、有历史组密钥存储机制等优点，适合文档管理系统的需求。针对基于分布式文档管理系统的新闻系统在访问控制上的不足,设计了基于区块链的访问控制方案。管理员签发的用户公钥与权限信息通过调用智能合约存储在区块链上，用户与作者端读取区块链上的用户公钥与权限信息,结合数字签名等机制,实现系统中的访问控制。设计并实现了基于区块链的文档管理系统，系统实现了文档管理系统的分布式运行，文档信息、个人公钥信息、组密钥信息、用户信息都存储在区块链上的状态数据库中,信息的录入、修改、删除等操作基于智能合约执行。系统基于区块链实现访问控制,对接入系统的用户进行身份认证,确保只有合法用户可以连接与访问资源。对保密文档进行加解密,实现保密文档存储、小组内部文档安全共享等功能,保证系统中的文档安全。

# 相关技术的说明

**区块链的介绍**

区块链是一项于2008年引入的新兴技术。它首先用作点对点分类账，用于注册比特币加密货币的交易。目的是消除任何第三方中介，并允许用户直接进行交易。为了实现这一目标，区块链被设计为对等节点的分散网络。网络中的每个节点具有以下特点：（1）持有交易分类帐的副本；（2）当它收到网络中其他节点的共识时，将一个条目写入其自己的分类帐；（3）将其用户进行的任何交易广播到网络中的其他节点，以及（4）定期检查其持有的分类帐是否与整个网络中的分类帐相同。随着比特币的持续普及，研究人员和实践者意识到其底层技术的巨大潜力。人们发现，区块链的独特功能（包括不变性，透明性和可信赖性）不仅在加密货币中有用，而且在许多其他领域中也有用。因此，已经开发了越来越多的基于区块链的应用程序在各个领域。目前，Rozeus是韩国上市集团BflySoft投资打造的去中心化内容体验平台，该项目将媒体、数字集体智慧、市民新闻、区块链融为一体，以此解决媒体产业主要问题并提供新的内容体验，

****基于区块链的应用程序的开发历程可以分为三个主要阶段：区块链1.0、2.0和3.0。区块链1.0用于加密货币，其重点是促进简单的现金交易。随后，针对房地产和智能合约引入了区块链2.0。这些智能合约规定了要在区块链中注册之前要满足的特定条件和标准。注册无需第三方干预即可进行。在Blockchain 3.0中，在政府，教育，卫生和科学等各个领域开发了许多应用程序。区块链在新闻鉴别的应用仍处于早期阶段。

IBM 商业价值研究院对 16 个国家或地区消费品行业中 203 个企业的高管进行了一项调研，其中包括零售企业和消费品 (CPG) 企业。我们发现，7% 的受访者希望在 2018 年大规模采用商用区块链解决方案。目前正在运用和投资区块链的受访者比例甚至更高，总计达到 18%。这些“先行者”希望区块链能够消除阻碍他们快速发展的摩擦。四分之三的受访者放眼新市场，69% 的受访者希望消除信息风险，64% 的受访者希望更好地适应监管环境。

“先行者”认为区块链的优势在以下六个方面表现得尤为突出：产品安全性和真实性、供应链优化、财务绩效、运营流程、促销策略管理以及客户互动与共创。在每个方面，他们所希望获得的不仅仅是节约时间和成本，或是降低风险等有目标的业务收益，而是发现有助于创造新业务模式或颠覆整个行业的机遇。无论起点如何，他们的终极目标都是扩大新的区块链解决方案的应用范围，覆盖到企业价值链的几乎每个方面。

“先行者”意识到，区块链所带来的机遇既存在于企业的供给侧，也存在于面向客户的互动端。他们可以利用区块链，动态地重新配置网络，进行实时优化，或者携手其他机构，获得更深入的消费者洞察。他们可以更有效地确保商品的安全和质量，也可以建立新的市场，转变自己看待和经营业务的方式。

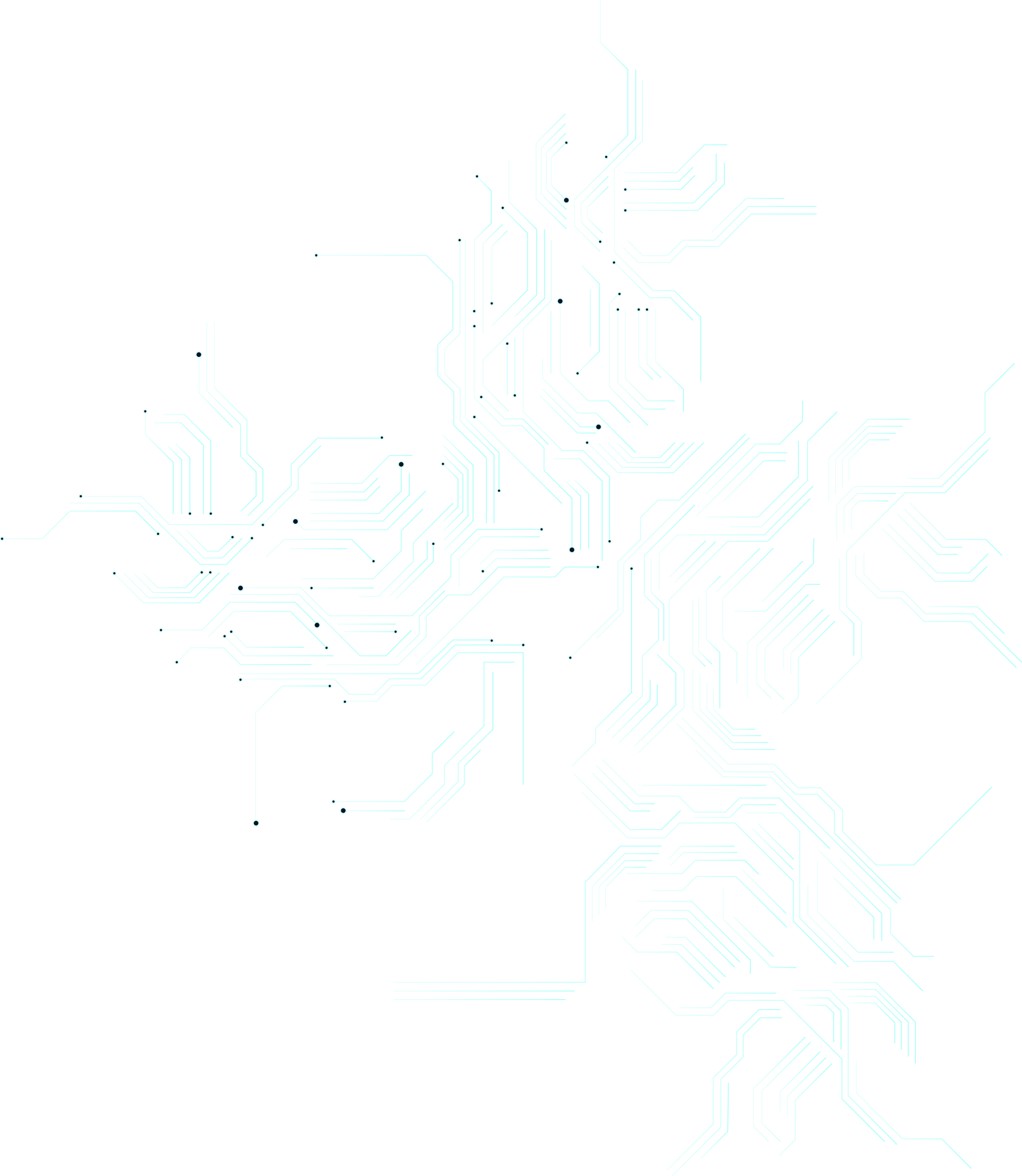
**智能合约**

基于区块链技术的智能合约不仅可以发挥智能合约在成本效率方面的优势，而且可以避免恶意行为对合约正常执行的干扰。将智能合约以数字化的形式写入区块链中，由区块链技 术的特性保障存储、读取、执行整个过程透明可跟踪、不可攥改。同时，由区块链自带的共识算法构建出一套状态机系统，使得智能合约能够高效地运行。数字形式意味着合约不得不写入计算机可读的代码中。这是必须的，因为只要参与方达成协定，智能合约建立的权利和义务，是由一台计算机或者计算机网络执行的。具体步骤分为：（1）**达成协定**（智能合约的参与方什么时候达成协定呢？答案取决于特定的智能合约实施。一般而言，当参与方通过在合约宿主平台上安装合约，致力于合约的执行时，合约就被发现了）。（2）**合约执行（**“执行”的真正意思也依赖于实施。一般而言，执行意味着通过技术手段积极实施）。（3）**计算机可读的代码（**另外，合约需要的特定“数字形式”非常依赖于参与方同意使用的协议）。

**数字签名**

数字签名（又称公匙）是只有信息的发送者才能产生的别人无法伪造的一段数字串，这段数字串同时也是对信息的发送者发送信息真实性的一个有效证明。它是一种类似写在纸上的普通的物理签名，但是使用了公钥加密领域的技术来实现的，用于鉴别数字信息的方法。一套数字签名通常定义两种互补的运算，一个用于签名，另一个用于验证。数字签名是非对称密钥加密技术与数字摘要技术的应用。主要特点：（1) 防冒充(伪造)。私有密钥只有签名者自己知道，所以其他人不可能构造出正确的。(2)可鉴别身份。在网络环境中，接收方必须能够鉴别发送方所宣称的身份。(3) 防篡改(防破坏信息的完整性)。而对于数字签名，签名与原有文件已经形成了一个混合的整体数据，不可能被篡改，从而保证了数据的完整性。(4)防重放。在数字名中，如果采用了对签名报文添加流水号、时间戳等技术，可以防止重放攻击。(5)防抵赖。在数字签名体制中，要求接收者返回一个自己签名的表示收到的报文，给对方或者第三方或者引入第三方机制。如此操作，双方均不可抵赖。(6)机密性(保密性)。有了机密性保证，截收攻击也就失效了。

**共识机制**

区块链的自信任主要体现于分布于区块链中的用户无须信任交易的另一方，也无须信任一个中心化的机构，只需要信任区块链协议下的软件系统即可实现交易。这种自信任的前提是区块链的共识机制，即在一个互不信任的市场中，要想使各节点达成一致的充分必要条件是每个节点出于对自身利益最大化的考虑，都会自发、诚实地遵守协议中预先设定的规则，判断每一笔记录的真实性，最终将判断为真的记录记入区块链之中。当节点们在网络中拥有公共信誉时，这一点体现得尤为明显。区块链技术正是运用一套基于共识的数学算法，在机器之间建立“信任”网络，从而通过技术背书而非中心化信用机构来进行全新的信用创造。现今区块链的共识机制可分为四大类：工作量证明机制、权益证明机制、股份授权证明机制和Pool验证池。（1）工作量证明机制：工作量证明机制即对于工作量的证明，是生成要加入到区块链中的一笔新的交易信息(即新区块)时必须满足的要求。在基于工作量证明机制构建的区块链网络中，节点通过计算随机哈希散列的数值解争夺记账权，求得正确的数值解以生成区块的能力是节点算力的具体表现。工作量证明机制具有完全去中心化的优点，在以工作量证明机制为共识的区块链中，节点可以自由进出。大家所熟知的比特币网络就应用工作量证明机制来生产新的货币。然而，由于工作量证明机制在比特币网络中的应用已经吸引了全球计算机大部分的算力，其他想尝试使用该机制的区块链应用很难获得同样规模的算力来维持自身的安全。同时，基于工作量证明机制的挖矿行为还造成了大量的资源浪费，达成共识所需要的周期也较长，因此该机制并不适合商业应用。（2）权益证明机制：权益证明机制根据每个节点拥有代币的比例和时间，依据算法等比例地降低节点的挖矿难度，从而加快了寻找随机数的速度。这种共识机制可以缩短达成共识所需的时间，但本质上仍然需要网络中的节点进行挖矿运算。因此，PoS机制并没有从根本上解决PoW机制难以应用于商业领域的问题。股份授权证明机制是一种新的保障网络安全的共识机制。它在尝试解决传统的PoW机制和PoS机制问题的同时，还能通过实施科技式的民主抵消中心化所带来的负面效应。。在这样的区块链中，全体节点投票选举出一定数量的节点代表，由他们来代理全体节点确认区块、维持系统有序运行。同时，区块链中的全体节点具有随时罢免和任命代表的权力。如果必要，全体节点可以通过投票让现任节点代表失去代表资格，重新选举新的代表，实现实时的民主。股份授权证明机制可以大大缩小参与验证和记账节点的数量，从而达到秒级的共识验证。然而，该共识机制仍然不能完美解决区块链在商业中的应用问题，因为该共识机制无法摆脱对于代币的依赖，而在很多商业应用中并不需要代币的存在。（3）PooI验证池：Pool验证池基于传统的分布式一致性技术建立，并辅之以数据验证机制，是目前区块链中广泛使用的一种共识机制。Pool验证池不需要依赖代币就可以工作，在成熟的分布式一致性算法(Pasox、Raft)基础之上，可以实现秒级共识验证，更适合有多方参与的多中心商业模式。

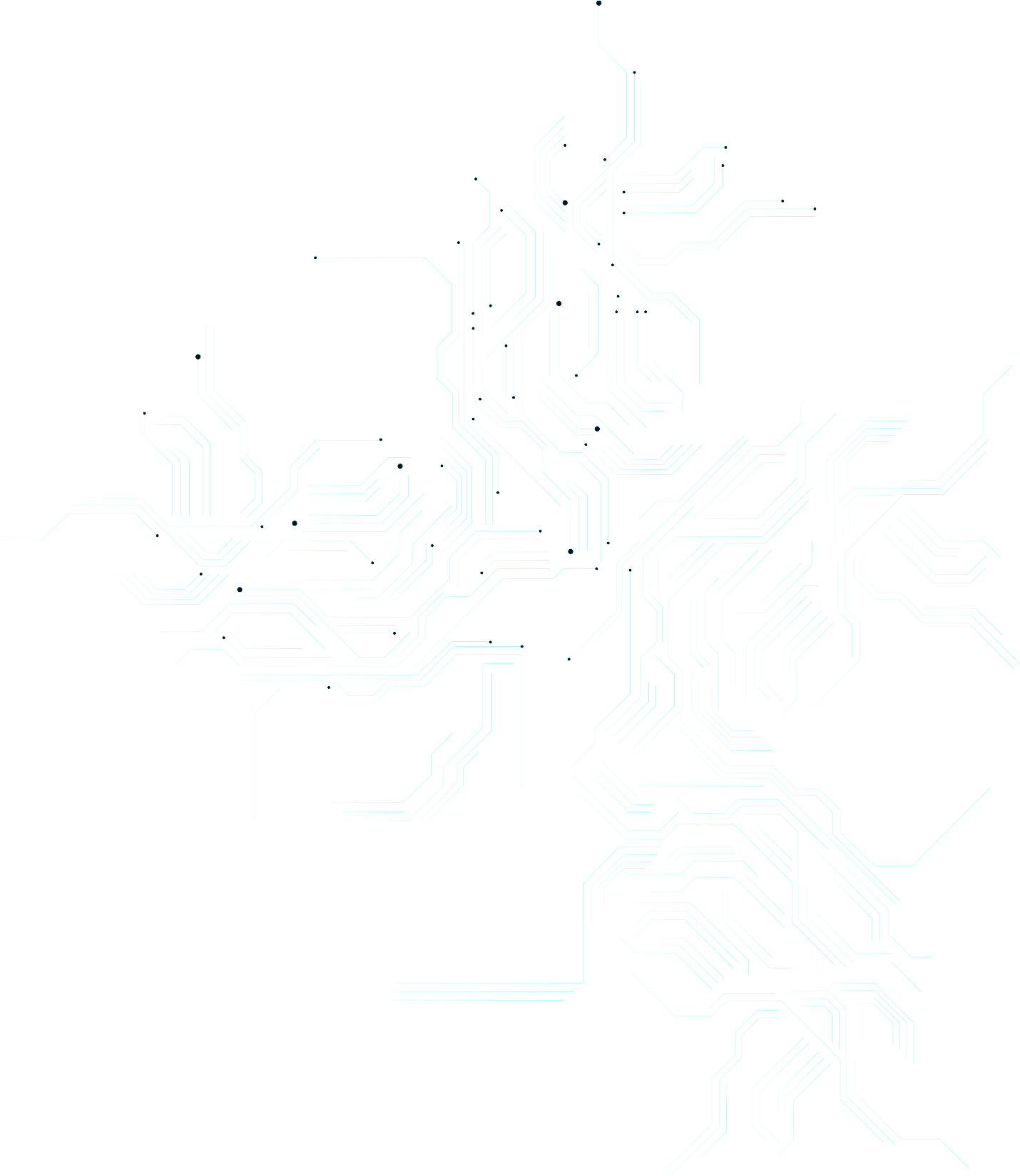
**超级账本**

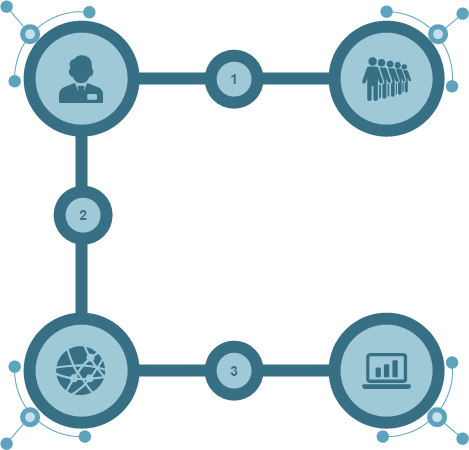
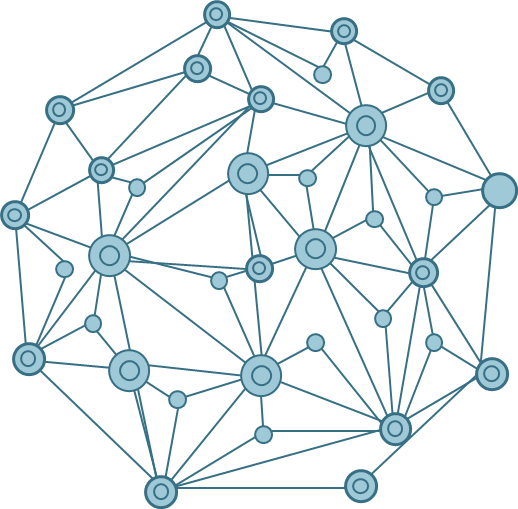
Hyperledger (或 Hyperledger项目)是一个旨在推动区块链跨行业应用的开源项目， 由 Linux基金会在2015年12月主导发起该项目， 成员包括金融，银行，物联网，供应链，制造和科技行业的领头羊。015年12月，Linux基金会宣布了Hyperledger项目的启动。创世成员于2016年的2月被宣布。随后的3月29日，另外的10名成员及治理理事会被公布.5月19日Brian Behlendorf被任命为项目的执行总监.

项目的目标是区块链及分布式记账系统的跨行业发展与协作，并着重发展性能和可靠性（相对于类似的数字货币的设计）使之可以支持主要的技术、金融和供应链公司中的全球商业交易。该项目将继承独立的开放协议和标准，通过框架方法和专用模块，包括各区块链的共识机制和存储方式，以及身份服务、访问控制和智能合约。

**密****钥协商协议**

随着通信系统的快速发展，密钥协商这一概念也得以提出。它指两个或多个实体协商，共同建立会话密钥，任何一个参与者均对结果产生影响，不需要任何可信的第三方会话密钥由每个协议参与者分别产生的参数通过一定的计算得出。**密钥协商协议的生成方式可分为证书型和无证书型：证书型**是指在会话密钥的产生过程中，由一个可信的证书中心（CA）给参与密钥协商的各方各分发一个证书，此证书中含有此方的公钥，ID及其他信息。证书型密钥协商协议的优点是提供认证，PKI（公钥密码体制）广泛部署，比较成熟，应用面广，且由PKG管理公私钥对有利于统一管理，缺点是计算代价大，需要一个可信的CA，同时证书还需要维护。**无证书型**是指各方在进行会话密钥的协商过程中不需要证书的参与，这是密钥协商协议的主流种类，优点是不需要CA的参与，减少了计算量，尤其是在低耗环境下应用的更多，同时安全性也不比证书型弱。几乎没有明显的缺点，只是设计一个安全的更加低耗的无证书密钥协商方案不是很容易。一方面它能为参与者提供身份认证，另一方面，也能与参与者协商并共享会话密钥。



整体框架设计  框架说明

三步规划

第三方审核员对新闻作者文章及二维码进行审核

**审核员**

每个新闻作者可在发布新闻时会自动生成相关的二维码

**新闻作者**

审核后的文章将发布在互联网上

**WEB**

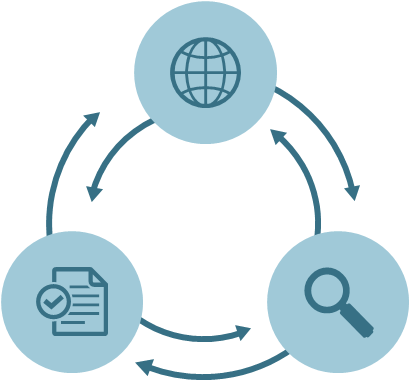
用户可以自由接收经过审核的新闻

**电脑**

1、新闻作者→审核员

新闻所采用的图片、照片，都需要提供有效的获取来源。同时，记者需要提供工作证明、电话地址、个人信誉等个人资料；新闻的出版社则需要提供其许可证、地址等材料。如果刻意制造假新闻，则当造谣实锤时，可根据信息对作者追究责任，无疑增大了造谣的成本。

**资料上传**



用户可以扫描二维码来追溯图片来源与作者信息等，判断新闻是否真实。作者的所有修改记录都将记录在区块中，如作者对已发布的新闻修改，修改的记录将不可篡改并可被所有人查看。

**防止篡改**

之前提交的材料都将由第三方核实，核实后的资料将会上传上链，并生成一个唯一的哈希值以及一个伴生的二维码，作者可以为了增加公信力来将二维码公布在新闻中。

**哈希生成**

2、审核员→互联网



审核员可以通过比特币类似的原理，对审核工作接单并担保，审核的时间和工作证明将保存在链上，审核员依据PoW机制收取报酬

**获利**

审核完毕后的新闻将携带作者的上传内容和审核员的审核记录上传到互联网

**上传**

审核员区别真假的判定标准

文本

审核员将对新闻的文本图片和证明进行人工或机器验证。其中着重查看证明

**审核**

文本

图片

证明

假图片

假证明

真新闻

假新闻

3、互联网→用户



区块链结合新闻筛选的新型新闻网站，将极大程度上提高新闻的真实性，以获得追求真实新闻的用户的青睐，从而逐渐起来旧有的杂乱新闻网站。而一旦非正规的新闻网站减少，假新闻也就失去了用户群体，失去了滋生的土壤。而二维码的衍生链接，也能吸引用户更多的流量。

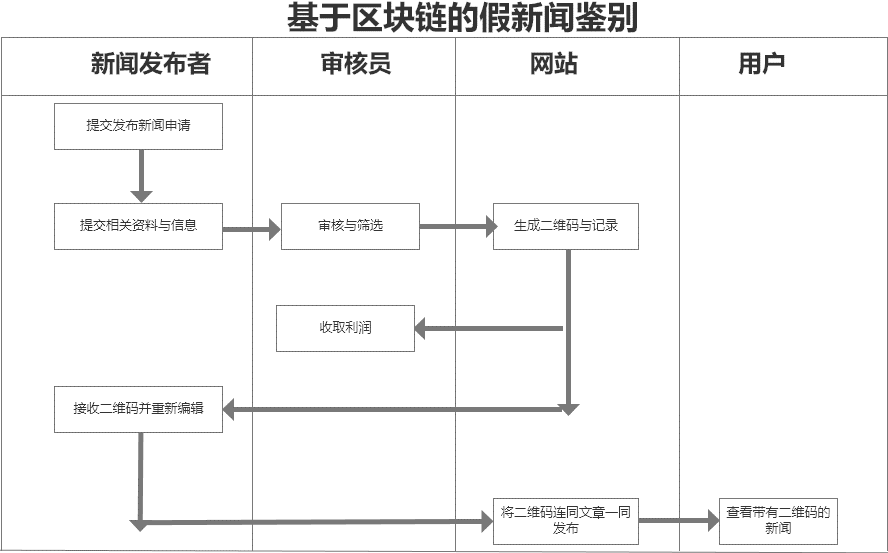
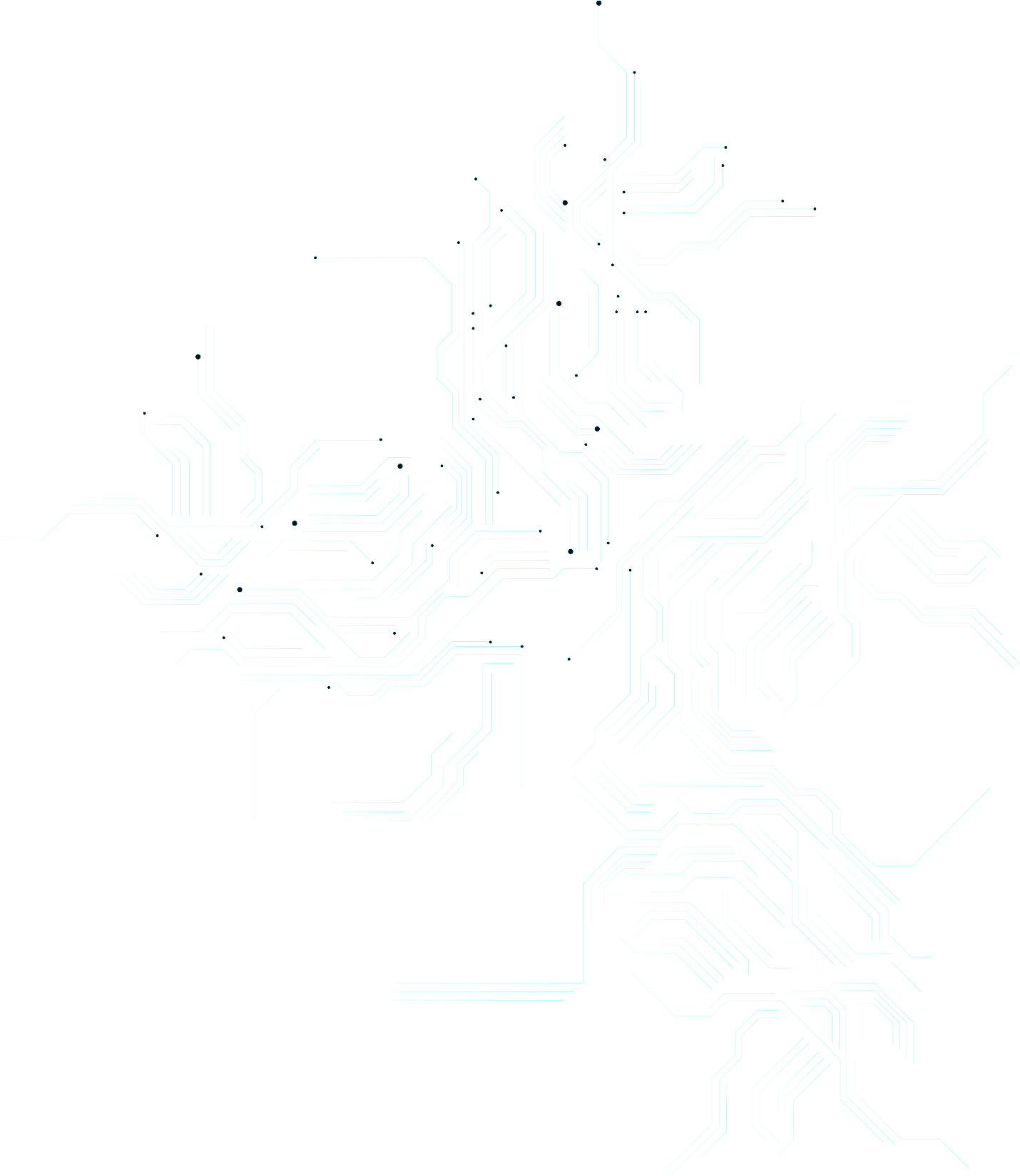
**互联网市场规划**

新闻网站可以仅上传携带二维码的新闻

未经过审核的新闻不携带官方二维码，用户可以选择不相信新闻

恶意造假生成假二维码的假新闻，扫描二维码后，用户可以通过查询区块链来验证真伪

4、结构导图



**基于区块链的密钥及文档管理系统方案**

## 密钥管理模块设计

初始化阶段：

1、各个Service首先在KMS中接入获得身份令牌token

2、各个Service生成自己的RSA公私钥

3、各个Service拿自己的RSA公钥去CA申请证书

密钥准备阶段：

1、Service用自己的证书去KMS申请需要的密钥

2、密钥保存在Service本地，定期去KMS重新获取（当有效期设置为0时，即不在Service本地保存，一次一密）

业务调用阶段：

1、Service用获取到的签名密钥做签名，加密密钥做加密，调用其他服务。

2、其他业务线校验签名，返回数据。

1、什么时候做数字签名？

每次接口调用都做数字签名。

2、数据加密的算法？

建议采用对称加密AES256位密钥，待加密的数据类型不同，选择不同模式，一般情况下CBC模式适合大多数场景，XTS模式适合本地存储场景。

3、如何判断一条数据是否被加密？

在系统迁移的过程中，必然出现明文和加密两种逻辑同事出现的情况，此时就需要程序判断数据是明文还是密文，建议在SDK中提供方法判断。

4、存储加密 数据库 索引如何处理？

基于安全的设计，相同的明文可能密文不同，因此需要建立一条不可逆并且与明文一一对应的值做索引。

5、存储加密历史数据如何处理？

第一次加密之前的历史数据，需要提前先由刷库 工具 统一将明文刷成密文，刷库时，需要先新建一列密文列，将明文列加密后刷到密文列中，之后程序写入或者更新操作时，需要对明文列、密文列双写，读取操作时只读取密文列，等程序稳定运行一段时间后，再将明文列删除。

第一次加密之后，随着密钥定期更新，不同时期的数据使用不同密钥加密。

6、密钥如何存储？

在KMS服务端，工作密钥需要加密存储于数据库中，加密存储的密钥可采用分段式密钥，通过RAID方式将不同密钥段存储于不同介质中。

7、证书如何生成下发？

证书生成下发通常有两种方式，第一种是SDK生成RSA公私钥，将RSA公钥发给CA申请证书，CA用自己的私钥签发证书后返回给SDK。第二种是直接CA生成RSA公私钥，并签发证书，并下发给SDK。这两种方式可根据实际业务需求选择。

证书是对客户端做身份识别的最重要标识，因此在第一次下发证书时，建议采用可信的第三方系统独立下发，如SRE发布系统。如果有有效且安全的手段保证客户端的合法性，可通过SDK与KMS的交互来下发证书。

8、证书如何验证，保证客户端的合法性？

证书验证需要两个步骤：

（1）验证证书的合法性，通过CA的公钥解密证书，校验签名即可验证证书的合法性、未被篡改。

（2）验证客户端持有证书对应的私钥，KMS向SDK发起challenge，向SDK发送一个随机数，SDK使用私钥加密后，返回给KMS，KMS使用证书中的公钥解密，验证SDK确实持有合法的私钥，证明SDK的合法身份。未了保证安全性，KMS发送的随机数可以做一次哈希。

9、密钥协商方式？

密钥协商可采用集中协商或者分散协商两种方式。

（1）集中协商：各个SDK分别向KMS请求密钥，KMS生成后返回给各个SDK。

（2）分散协商：假设有两个客户端A和B，A和B使用DH密钥协商算法，来协商密钥。

## 文档管理模块设计

1. 概要设计的目的主要采用比较抽象的语言进行总结，对整个需求确定系统的物理配置，并确定整个系统的处理流程和系统的数据结构，以及界面设计，对系统进行初步设计。确定系统的物理配置,确定整个系统的处理流程和数据结构的系统,界面设计,和实施系统初步设计

SQLSever服务器:数据库管理软件

SQL:是一种数据库查询和程序设计语言

DBMS:使用一个数据库查询和程序设计语言

JDBC:是一个Java 语句来执行SQL语句,

1. 整体设计

规定需求：系统实现安装方便,易维护,易操作性,运行稳定,安全可靠。舞台是为了明确系统的数据结构和软件结构,除了总体设计还将给出内部和外部软件系统组件之间的接口定义,每个软件的功能模块,数据结构和特定需求的细节。

时间特性要求：

本系统要求程序在响应时间、数据更新处理时间上性能较好。而且它也由于数据量比较小，所以数据传输的时间也可以准时和系统运行的时间。

接口设计：

用户界面：使用窗口，菜单设计，操作时响应一个热键。由用户在输入窗口中输入登录名和密码进入各个模块。

外部接口：支持普通电脑或笔记本电脑:软件应提供对条形码扫描仪的支持等。通过JDBC连接SQL Server数据库。

内部接口：1、通过面向对象的语言设计类，实现公共类中的调用;对类之间进行严格的封装2，模块为文档管理系统提供操作符和系统参数等基础数据。操作符必须设置为使用其他模块。3、文档管理模块为文档统计模块、文档查询模块提供基本数据和文档。必须有文档数据后，才能使用统计和查询模块。4、文档管理模块和文档证处理模块为文档管理模块提供数据基础。

## 用户管理模块设计

用户管理模块提供对用户信息的管理，包括用户注册、用户登录、用户权限管理、用户信息修改以及用户等级修改。

**用户注册**

根据用户表，设计相应的注册页面，注册页面包括用户名、密码、邮箱、部门、电话等信息，当用户进行注册时，填写这些信息，用户名是不能与已注册的

用户名相同，填写完成后，提交注册请求，后台相应的Action会响应该动作,首先获取到页面发来的参数然后将这些参数通过Session

对象写入到数据库中，最后向用户提示注册成功与否。

**用户登录**

用户注册之后，就可以通过账户和密码登陆至平台。当用户提交登陆请求，后台相应Action的会响应该动作，首先获取到页面发来的用户名和密码，然后通过Query

对象查询该用户是否存在且密码正确，最后将根据结果给用户发送跳转页面，如果用户存在且密码正确，则可进入平台主页面，否则，提示登陆错误

信息。

用户权限管理

用户权限管理将用户分为普通用户和管理员，他们具有不同的权限，自的权限如表所示。此平台首次使用时，会内置一个超级管理员，有修改用户等级的权限。

## 智能合约设计

智能合约的设计与选型

优秀的智能合约设计应该是场景驱动的。严格来说，智能合约不算是基础设施，而更像开发框架。既然是开发框架，其设计目标应该是纯粹的——以最低成本、最简单的形式全力解放目标领域的生产力。

例如，比特币区块链专注于货币交易，其脚本语言也非常简单，且是非图灵完备的；以太坊与EOS目标是通用的信任基础设施，故设计了图灵完备、开发体验良好的智能合约方案；Zilliqa目标是提供公开平台给需要高可扩展性计算资源的应用，如数据挖掘、机器学习、金融模型等，场景比较垂直，其合约语言Scilla也设计成非图灵完备的，能充分满足领域需求，易于形式化验证，且语法更简洁；Libra目标是提供安全稳定的基础设施运行其稳定币，其智能合约Move将资产提升为first-class，不得随意复制、创建，提升了编写难度，但从语言层面保证了资产的一致性。

安全性(Security)

智能合约编写，首要考虑的就是安全性问题。

在区块链世界中，恶意代码数不胜数。如果你的合约包含了跨合约调用，就要特别当心，要确认外部调用是否可信，尤其当其逻辑不为你所掌控的时候。

如果缺乏防人之心，那些“居心叵测”的外部代码就可能将你的合约破坏殆尽。比如，外部调用可通过恶意回调，使代码被反复执行，从而破坏合约状态，这种攻击手法就是著名的Reentrance Attack（重放攻击）。

这里，先引入一个重放攻击的小实验，以便让读者了解为什么外部调用可能导致合约被破坏，同时帮助更好地理解即将介绍的两种提升合约安全性的设计模式。

例如AddService合约是一个简单的计数器，每个外部合约可以调用AddService合约的addByOne来将字段\_count加一，同时通过require来强制要求每个外部合约最多只能调用一次该函数。

这样，\_count字段就精确的反应出AddService被多少合约调用过。在addByOne函数的末尾，AddService会调用外部合约的回调函数notify。AddService的代码如下：

contract AddService{

uint private \_count;

mapping(address=>bool) private \_adders;

function addByOne() public {

//强制要求每个地址只能调用一次

require(\_adders[msg.sender] == false, "You have added already");

//计数

\_count++;

\_adders[msg.sender] = true;

//调用账户的回调函数

AdderInterface adder = AdderInterface(msg.sender);

adder.notify();

//将地址加入已调用集合

\_adders[msg.sender] = true;

}

}

contract AdderInterface{

function notify() public;

}

由于\_adders映射已经修改完毕，当恶意攻击者想递归地调用addByOne，require这道防线就会起到作用，将恶意调用拦截在外。

**可维护性（Maintaince）**

在区块链中，合约一旦部署，就无法更改。当合约出现了bug，通常要面对以下问题：

1. 合约上已有的业务数据怎么处理？
2. 怎么尽可能减少升级影响范围，让其余功能不受影响？
3. 依赖它的其他合约该怎么办？

回顾面向对象编程，其核心思想是将变化的事物和不变的事物相分离，以阻隔变化在系统中的传播。所以，设计良好的代码通常都组织得高度模块化、高内聚低耦合。利用这个经典的思想可解决上面的问题。

**Data segregation - 数据与逻辑相分离**

了解该设计模式之前，先看看下面这个合约代码：

contract Computer{

uint private \_data;

function setData(uint data) public {

\_data = data;

}

function compute() public view returns(uint){

return \_data \* 10;

}

}

此合约包含两个能力，一个是存储数据(setData函数)，另一个是运用数据进行计算(compute函数）。如果合约部署一段时间后，发现compute写错了，比如不应是乘以10，而要乘以20，就会引出前文如何升级合约的问题。

这时，可以部署一个新合约，并尝试将已有数据迁移到新的合约上，但这是一个很重的操作，一方面要编写迁移工具的代码，另一方面原先的数据完全作废，空占着宝贵的节点存储资源。

所以，预先在编程时进行模块化十分必要。如果我们将"数据"看成不变的事物，将"逻辑"看成可能改变的事物，就可以完美避开上述问题。Data Segregation（意为数据分离）模式很好地实现了这一想法。

该模式要求一个业务合约和一个数据合约：数据合约只管数据存取，这部分是稳定的；而业务合约则通过数据合约来完成逻辑操作。我们将数据读写操作专门转移到一个合约DataRepository中：

contract DataRepository{

uint private \_data;

function setData(uint data) public {

\_data = data;

}

function getData() public view returns(uint){

return \_data;

}

}

计算功能被单独放入一个业务合约中：

contract Computer{

DataRepository private \_dataRepository;

constructor(address addr){

\_dataRepository =DataRepository(addr);

}

//业务代码

function compute() public view returns(uint){

return \_dataRepository.getData() \* 10;

}

}

这样，只要数据合约是稳定的，业务合约的升级就很轻量化了。比如，当我要把Computer换成ComputerV2时，原先的数据依然可以被复用。

**生命周期（Lifecycle）**

在默认情况下，一个合约的生命周期近乎无限——除非赖以生存的区块链被消灭。但很多时候，用户希望缩短合约的生命周期。这一节将介绍两个简单模式提前终结合约生命。

**Mortal - 允许合约自毁**

字节码中有一个selfdestruct指令，用于销毁合约。所以只需要暴露出自毁接口即可：

contract Mortal{

//自毁

function destroy() public{

selfdestruct(msg.sender);

}

}

**权限（Authorization）**

前文中有许多管理性接口，这些接口如果任何人都可调用，会造成严重后果，例如上文中的自毁函数，假设任何人都能访问，其严重性不言而喻。所以，一套保证只有特定账户能够访问的权限控制设计模式显得尤为重要。

**Ownership**

对于权限的管控，可以采用Ownership模式。该模式保证了只有合约的拥有者才能调用某些函数。首先需要有一个Owned合约：

contract Owned{

address public \_owner;

constructor() {

\_owner = msg.sender;

}

modifier onlyOwner(){

require(\_owner == msg.sender);

\_;

}

}

如果一个业务合约，希望某个函数只由拥有者调用，该怎么办呢？如下：

contract Biz is Owned{

function manage() public onlyOwner{

}

}

这样，当调用manage函数时，onlyOwner修饰符就会先运行并检测调用者是否与合约拥有者一致，从而将无授权的调用拦截在外。

**行为控制（Action And Control）**

这类模式一般针对具体场景使用，这节将主要介绍基于隐私的编码模式和与链外数据交互的设计模式。

**Commit - Reveal - 延迟秘密泄露**

链上数据都是公开透明的，一旦某些隐私数据上链，任何人都可看到，并且再也无法撤回。

Commit And Reveal模式允许用户将要保护的数据转换为不可识别数据，比如一串哈希值，直到某个时刻再揭示哈希值的含义，展露真正的原值。

以投票场景举例，假设需要在所有参与者都完成投票后再揭示投票内容，以防这期间参与者受票数影响。我们可以看看，在这个场景下所用到的具体代码：

contract CommitReveal {

struct Commit {

string choice;

string secret;

uint status;

}

mapping(address => mapping(bytes32 => Commit)) public userCommits;

event LogCommit(bytes32, address);

event LogReveal(bytes32, address, string, string);

function commit(bytes32 commit) public {

Commit storage userCommit = userCommits[msg.sender][commit];

require(userCommit.status == 0);

userCommit.status = 1; // comitted

emit LogCommit(commit, msg.sender);

}

function reveal(string choice, string secret, bytes32 commit) public {

Commit storage userCommit = userCommits[msg.sender][commit];

require(userCommit.status == 1);

require(commit == keccak256(choice, secret));

userCommit.choice = choice;

userCommit.secret = secret;

userCommit.status = 2;

emit LogReveal(commit, msg.sender, choice, secret);

}

}

## 客户端设计

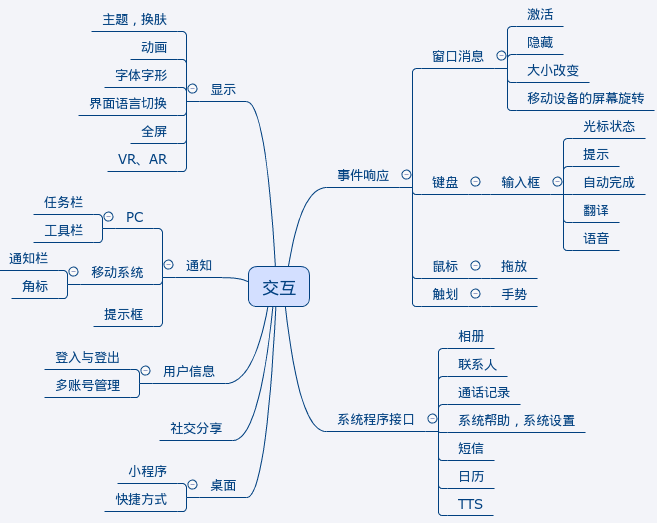
**1 基础设施**

程序最基本的处理就是数据IO以及为了并行计算所做的操作，它们通常会作为程序的底层框架，供上层使用。



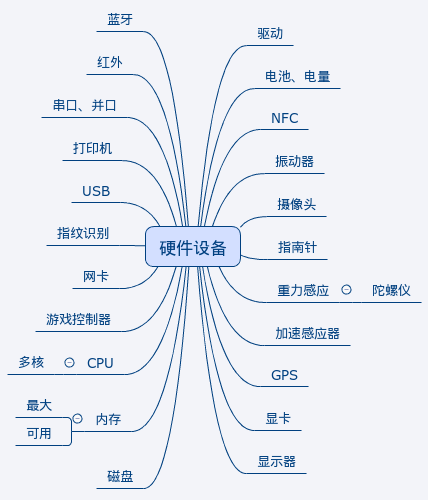
**2 交互**

从PC到现在的移动设备还有VR，人机交互中的输入设备在不断自然化，可是展示界面的元素仍然还是这几种：静态文本、按钮、列表、输入框、选择菜单、进度提示、滚动条、标签页。每种UI框架都提供这些基本组件，可以说掌握它们就等于对这个框架入门了。高级的交互会跨越窗口、跨越程序，并能更智能化地响应输入。



**3 硬件设备**

PC上的硬件大都需要通过统一的接口来操作，开发者需要做好繁复的参数设置才能正确使用。移动系统则对很多硬件做了封装，为开发者提供定制的简单API。有时候开发者并不需要直接操控硬件，但需要获取硬件的信息，根据可用的资源数来执行不同的策略。



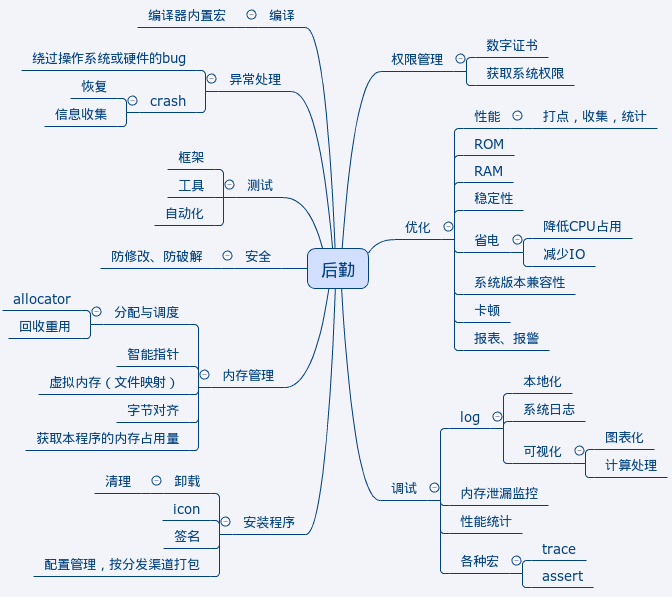
**4 程序设计**

现在的程序流行快速迭代，没过几天就发布一个新版本。新的app还要越快上线越好，尽早占据制高点。如此匆忙的节奏，让传统软件的架构设计退居二线，程序往往都是技术债务多到影响开发速度或新功能才执行大的重构优化。不过我们至少可以提前根据功能分出一些模块，让后面的重构稍微轻松些。



**5 后勤**

如果说需求开发是前线，那么产品经理无暇关心的东西都算后勤了。这部分工作需要技术经理主导来开展。对程序员来说这部分工作可能没那么刺激，但却是深入学习系统和各种工具原理的好机会。



**基于区块链的文档管理系统实现与测试**

## **系统开发环境及配置**

|  |  |
| --- | --- |
| 计算机硬件 | * 需要 64 位处理器和操作系统 * 计算机型号：OMEN by HP Laptop 15 * 操作系统: Windows 10 - April 2018 Update (v1803) * 处理器: Intel® Core™ i7-9750H * 显卡: Nvidia GeForce GTX 1660Ti 6GB |
| 软件 | Remix ide、Google Chrome、python |
| 其它 | **网络:** 宽带互联网连接 |

## **环境搭建**

区块链-创世区块（windows私链以太坊网络搭建）

1.作为如今最大的区块链开源平台，必须要安装 以太坊

Geth windows官方安装指南：

https://github.com/ethereum/go-ethereum/wiki/Installation-instructions-for-Windows（全英文的，建议用gogle浏览器，直接右键翻译）

ps：安装的时候在c盘中找到cmd.exe 右键利用管理员权限安装，因为需要安装许多安装包，不然会报错

2. 区块链白皮书地址：https://github.com/ethereum/wiki/wiki/White-Paper

3.因为geth需要同步所有区块，所以时间较长， 大概要一天时间，

4.以上按照官方文档操作即可，都准备好后，建立创世纪区文件，为json格式，然后在json文件里面填入新区块配置:

{

"config": {

"chainId": 15,

"homesteadBlock": 0,

"eip155Block": 0,

"eip158Block": 0

},

"coinbase" : "0x0000000000000000000000000000000000000000",

"difficulty" : "0x40000",

"extraData" : "",

"gasLimit" : "0xffffffff",

"nonce" : "0x0000000000000042",

"mixhash" : "0x0000000000000000000000000000000000000000000000000000000000000000",

"parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",

"timestamp" : "0x00",

"alloc": { }

}

解释一下各个参数的作用：

Mixhash：与nonce配合用于挖矿，由上一个区块的一部分生成的hash。注意他和nonce的设置需要满足以太坊的Yellow paper, 4.3.4. Block Header Validity, (44)章节所描述的条件。.

Nonce：nonce就是一个64位随机数，用于挖矿，注意他和mixhash的设置需要满足以太坊的Yellow paper, 4.3.4. Block Header Validity, (44)章节所描述的条件。

Difficulty：设置当前区块的难度，如果难度过大，cpu挖矿就很难，这里设置较小难度

Alloc：用来预置账号以及账号的以太币数量，因为私有链挖矿比较容易，所以我们不需要预置有币的账号，需要的时候自己创建即可以。

Coinbase：矿工的账号

Timestamp：设置创世块的时间戳

parentHash：上一个区块的hash值，因为是创世块，所以这个值是0

extraData：附加信息

gasLimit：该值设置对GAS的消耗总量限制，用来限制区块能包含的交易信息总和，因为我们是私有链，所以填最大。

5.启动区块链节点：

进入windows的以太坊目录下，创建文件夹chain，存放区块资料，然后把之前创建好的json文件放在里面，

piccgenesis.json即为创世区块配置

6.a.进入geth安装目录：C:\Users\admin\AppData\Roaming\Ethereum

b.执行：geth --datadir "%cd%\chain" init piccgenesis.json

启动Geth即可以启动以太坊的区块链，为了构建私有链 ，需要在Geth启动时加入一些参数，指令如下：

geth --identity "PICCetherum" --rpc --rpccorsdomain "\*" --datadir "%cd%\chain" --port "30303" --rpcapi "db,eth,net,web3" --networkid 93318 console

执行成功后，会进入javascript 控制台：以太坊私有网络搭建成功

Geth参数含义如下：

identity区块链的标示

Init：指定创世块文件的位置，并创建初始块

Datadir：设置当前区块链网络数据存放的位置

Port：网络监听端口

Rpc：启动rpc通信，可以进行智能合约的部署和调试

Rpcapi：设置允许连接的rpc的客户端，一般为db,eth,net,web3

Networkid：设置当前区块链的网络ID，用于区分不同的网络，是一个数字

Console：启动命令行模式，可以在Geth中执行命令

7.进入的JS控制台后使用节点创建账号：

a.输入指令查看账号信息：指令：eth.accounts

正常情况下是没有的 下面的输出为[]，因为之前我创建过，所以我才有账号，

b.创建账户：指令：personal.newAccount("wangyaxing")，

d.再次输入查询账号指令：eth.accounts

可见我有两个账号以数组形式存放

8.账户已经有了，而且这个区块链使我们自己创建的，里面的货币数量是也我们定义的，

赚钱方法英文叫Mining 俗称挖矿

a. 在命令行 执行开始挖矿：指令：miner.start()

因为是私有节点，并且没有竞争者，所以我们一分钟 就会有获得很多的奖励，

输入指令：miner.stop()，停止挖矿

查询钱包余额指令：eth.getBalance(eth.accounts[0])

查询的是第一个账户的余额，当我们挖矿时，奖励金存储默认的是第一个账户，如果第二个账户需要，我们可以执行转账指令；

 eth.sendTransaction({from:"第一个账户的账号",to:"第二个账户的账号",value:web3.toWei(1000000,"ether")})

当直接执行此方法时会抛出异常：

account is locked

    at web3.js:3119:20

    at web3.js:6023:15

    at web3.js:4995:36

    at <anonymous>:1:1

     是没解锁的原因

解锁指令：

personal.unlockAccount("第一个账户的账号","设定的密码")

然后需要执行挖矿命令，才会把转账真正完成，会看到转的账户数字增加。

## 二维码+区块链的实现

**二维码与区块链应用场景天然契合**

如今无论是在虚拟世界的各类网页、APP，还是在现实世界的便利店、地铁、户外广告、杂志上，随处可见小小的方形花纹图案，用户取出手机“扫一扫”的一瞬间，即可完成信息获取、电子交易以及二维码凭证获取。随着互联网、移动互联网、物联网的迅速发展，二维码作为重要的信息入口，在物联网、供应链、医疗、版权、教育、共享经济等领域与区块链应用场景具有诸多契合点。

**区块链借助二维码****推动传统行业转型升级的背后逻辑**

区块链具有去中心化、不可篡改、共享账本、不对称加密、智能合约等功能，可以解决多个领域应用的实际痛点。譬如去中心化特性在多方参与的业务流程中可以绕开中心机构，降低中间成本；不可篡改的特点可以确保链上信息的真实性；共享账本构建信任基础保障业务透明可追溯；不对称加密可以保障信息安全；智能合约支持自动交易提升系统运转效率。

当前，区块链尚未落地应用的核心问题是区块链的“硬链接”难题，即在区块链内的数字资产与区块链外的实物或虚拟资产之间难以建立起牢固的、可信任的链接绑定关系。二维码效率高、成本低、使用方便，无论是金融、物流、供应链，还是制造、出版、物联网等产业，二维码为各类信息上链提供了极为便捷的入口，有望解决区块链上资产与链下实物锚定的关键问题，成为未来数字社会区块链上信息、资产与链下实物锚定的重要工具。

**案例：“区块链+二维码”在防伪溯源领域的应用**

传统供应链的溯源防伪系统存在信息不透明、数据容易篡改、安全性差、相对封闭等弊端，近年来假冒伪劣产品扰乱了良好的市场经济秩序，对人民群众生活消费和工农业生产带来极大损害。当前IBM、京东、菜鸟、智链、唯链等企业已开展“区块链+二维码”在溯源防伪领域的落地应用，如京东利用二维码技术和区块链技术，有效打通供货商、销售商、物流、政府、检测机构间的防伪、溯源、全程追溯信息，将商品原材料采买过程、生产过程、流通过程、营销过程的信息进行整合和追溯，真正实现跨越品牌商、渠道商、零售商、消费者，精细到一物一码（或一批次一码）的全流程正品追溯，并通过多个网络渠道全面展示给消费者，显著提升了用户信任体验，同时也为企业供应链全流程信息管理提供了有力的安全保障，实现了企业和消费者收益的最大化。

**如何利用网络资源制作二维码+区块链**

目前，有草料二维码等一大批网站支持二维码的生成与定制，包括企业级二维码定制。



## 系统功能测试

**内容**

主要内容包括功能测试和健壮性测试

**功能测试**

功能测试也叫黑盒测试或数据驱动测试，只需考虑需要测试的各个功能，不需要考虑整个软件的内部结构及代码，一般从软件产品的界面、架构出发，按照需求编写出来的测试用例，输入数据在预期结果和实际结果之间进行评测，进而提出更加使产品达到用户使用的要求。

**健壮性测试**

健壮性测试测试软件在输入错误、磁盘故障、网络过载或有意攻击情况下，能否不死机、不崩溃

**常见系统测试分类**

**恢复测试**

恢复测试指持续超过系统规格负载测试之后，再将负载恢复到规格以内的测试方法，同时，恢复测试还关注导致软件运行失败的各种条件，并验证其恢复过程能否正确执行。在特定情况下，系统需具备容错能力。另外，系统失效必须在规定时间段内被更正，否则将会导致严重的经济损失。

**安全测试**

安全测试用来验证系统内部的保护机制，对信息、数据的保护能力，以防止非法侵入。在安全测试中，测试人员扮演试图侵入系统的角色，采用各种办法试图突破防线。因此系统安全设计的准则是要想方设法使侵入系统所需的代价更加昂贵

**压力测试**

压力测试指一段时间内持续超过系统规格的负载进行测试的一种可靠性测试方法

**系统测试流程**

* 测试计划：测试小组协商后又组长起草测试计划，计划内容包含测试的范围、方法、环境工具、测试完成标准以及测试人员分配
* 测试设计：即设计测试用例，测试小组根据测试计划安排，参考测试需求、软件规格说明书以及其他资料、设计合适的测试用例
* 测试执行：测试成员按照测试计划、测试用例设计执行测试，并得到记录结果
* 测试结果管理：管理软件缺陷，向开发反馈，纠正软件缺陷

**项目检查表**

|  |  |
| --- | --- |
| 检查项 | 测试人员的类别及其评价 |
| 窗口切换、移动、改变大小时正常吗？ | 是 |
| 各种界面元素的文字正确吗？（如标题、提示等） | 是 |
| 各种界面元素的状态正确吗？（如有效、无效、选中等状态） | 是 |
| 各种界面元素支持键盘操作吗？ | 是 |
| 各种界面元素支持鼠标操作吗？ | 是 |
| 对话框中的缺省焦点正确吗？ | 是 |
| 数据项能正确回显吗？ | 是 |
| 对于常用的功能，用户能否不必阅读手册就能使用？ | 是 |
| 执行有风险的操作时，有“确认”、“放弃”等提示吗？ | 是 |
| 操作顺序合理吗？ | 是 |
| 有联机帮助吗？ | 是 |
| 各种界面元素的布局合理吗？美观吗？ | 是 |
| 各种界面元素的颜色协调吗？ | 是 |
| 各种界面元素的形状美观吗？ | 是 |
| 字体美观吗？ | 是 |
| 图标直观吗？ | 是 |
|  |  |
| … |  |

# 总结

软件工程的文档是根据软件的生存周期产生的，从软件的产生到软件的报废这一过程中，不仅需要系统程序的运行使用，还要有软件开发文档的密切指导，为软件生存周期的运转提供前进和完善的方向，都说软件工程是软件开发的灵魂，那么软件工程中的开发文档便是灵魂思想的指引者，软件危机的产生很重要的原因是缺乏一整套完整的开发文档，根据我个人的理解，开发文档对于软件来说是软件前进的道路，如果软件开发缺少了文档的指引，那么软件即使开发出来也是漫无方向的原地踏步，对于软件开发的工程化道路一定要要把软件开发文档的编写重视起来。