量子链
Quantum

QUANTUM WHITE PAPER

Value Transfer Protocol and DAPP Platform

量子链白皮书

价值传输协议及去中心化应用平台

foundation@qtum.org
量子链基金会

2016 年 8 月 1 日

目录

摘要 Abstract

Quantum Blockchain （简称**‘Qtum’** [ˈkwɒntəm]，**量子链**）致力于开发比特币和以太坊之外的第三种区块链生态系统，并致力于拓展区块链技术的**应用边界和技术边界**，使**普通互联网用户**能感受到区块链技术的价值。

在 Qtum 系统中，可以通过价值传输协议（Value Transfer Protocol）来实现点对点的价值转移，并根据此协议，构建一个支持多个行业的（金融、物联网、供应链、社交游戏等）去中心化的应用开发平台（DAPP Platform）。

Qtum 系统通过良好的设计原则和设计策略来实现，例如兼容性原则、模块化设计策略、安全性策略和易用性策略。 从技术角度分析，Qtum 致力于实现**首个兼容 BIP（基于 UTXO 模型 ）的 POS 智能合约平台，并通过 Identity，Oracle 和 Data feeds 的引入。在合规性方面，符合不同行业的监管需求。**在 Qtum 的公链（Public Blockchain）系统中，在共识机制上，从去中心化程度、实用性、技术可靠性考虑，我们将以 Proof of Stake 为基础，加入节点在线激励因素（Incentive Factor），形成 IPOS（Incentive POS）的共识协议。在 Qtum 的联盟链中（Permissioned Blockchain），我们将采用 Qtum 开发者提出的与 Raft 融合的 Proof of Time 共识协议，使得在联盟链或者私链中，达成共识的时间大大缩短（BlockTime:250ms，Confirmation Time：750ms-3s）。

Qtum 系统将基于 UTXO 模型来实现基于区块链的合约，主要考虑以下因素：（1）与比特币生态的兼容性； （2）BIP 长期演进协议的兼容性 ；（3）交易的并行处理能力/隐私性/可追溯性。

在 Qtum 系统中，我们把**区块链合约**（Blockchain Contract）分成 **智能合约**（Smart Contract）和**主控合约**（Master Contract），除了支持智能合约外，我们将通过链下因素的引入，形成符合现实世界商业逻辑的区块链主控合约。另外在虚拟机方面，在 Qtum 的测试网络中，我们将兼容 EVM，后期通过标记不同的虚拟机类型，可以支持更多的虚拟机，包括 LLVM 和 Lua 以及 EVM2.0. 以及为 VM 开发的更严格的编程语言。

在 Qtum 系统中，我们通过 Oracle 和 Data Feed 的设计，可以让区块链的智能合约更落地和更符合商业规则，搭建了现实世界到区块链世界的桥梁。另外 Qtum 系统中，可以通过智能合约来管理参与者的身份信息，将为基于 Qtum 系统的金融服务提供更好的支持。

最后面向移动端策略（**Go Mobile**）也是 Qtum 特别重视的一个战略，在量子链的生态系统中，我们将会与第三方开发者，一起从技术架构支持提供移动端的服务，包括：移动端钱包、移动端 DAPP 应用、移动端智能合约服务。 我们也鼓励第三方的开发者，加入我们，一起开发区块链的移动端服务，共同推动区块链技术的落地。

量子链白皮书
价值传输协议及去中心化应用平台

帅初 shuaichu@qtum.org

第一部分 量子链设计原则和理念 Principles

1.1 区块链出现的背景和意义

在 2008 年 10 月 31 日，Satoshi Nakamoto <satoshi@vistomail.com> 通过 一个密码学小组（gmane.comp.encryption.general）发送了一封邮箱，并第一次公布了比特币的白皮书《Bitcoin: A Peer to Peer Electronic Cash System》，并提出了比特币网络的一些特点：

1. Double-spending is prevented with a peer-to-peer network 防止双花
2. No mint or other trusted parties 无铸币厂或其他信任方
3. Participates can be anonymous 参与者可匿名
4. New coins are made from Hashcash style Proof-of-work 通过工作量证明方式发行新币
5. The proof-of-work for new coin generation also powers the network to prevent double-spending 基于工作量证明的新币发行过程中，也同时阻止了双花的发生

在 2009 年 1 月 3 号，比特币的创始区块被挖出，并在第 170 个区块发生了第一笔比特币的转账交易（从 Satoshi 到 Hal Finney，发生在 2009 年 1 月 12 号），从此开启了比特币网络作为一种点对点的价值交换网络蓬勃发展的时代，虽然中间经历了各种危机，但是比特币网络的价值从零开始，到今天已经成为一个价值约 100 亿美金的点对点支付网络。

点对点价值传输网络的出现有其历史必然性，而 Satoshi 则是加速这个历史进程的人。从上个世纪 80 年代，TCP/IP 协议的开发，到 90 年代，网页浏览器的应用和服务器的应用，一直到今天，互联网技术从不同侧面和维度改变了数据交换的模式和人类的生活。互联网技术的发展得益于基础设施的完善，从早期的 信息高速公路（Information Super Highway ）和各种智能终端的普及，这些也构成了互联网 OSI 七层模型中，应用层无限拓展的基础。

在互联网的各种协议栈中，我们用的较多有 TCP/IP，HTTP，HTTPS，FTP，TELNET，SSH，SMTP，POP3 等网络层，传输层，应用层的协议，并且借助这些协议，我们已经比较完美了搭建了各种各样的互联网服务。但是如果我们深思，我们会发现，在比特币网络出现之前，我们一直无法互联网上面，在不借助于第三方的情况下，较好的进行点对点的价值的转移和传输。其实我们并不是缺少一种特定的方法，而是缺少基于信息高速公路（Information Super Highway）的价值高速公路（Value Super Highway），以及如何实现 **Value Super Highway** 的 **Value Transfer Protocol** （VTP 协议），而比特币网络则是运行于 信息高速公路上面的第一个 VTP 协议。在量子链的白皮书中，我们也第一次归纳和提出了互联网应用层 Value Transfer Protocol 的概念。

随着互联互通技术的发展（互联网、物联网、VR/AR），人与物体、人与信息的交互方式更加多样化，更多的实体被数字化（Digitalize）和令牌化或者代币化（Tokenize）和符号化（Symbolize），一旦实体被数字化或者代币化之后，就完成了实体资产在互联网上面的映射和切分，马上面临的一个问题就是：如何点对点传输这些资产和价值？

因此可以推测，随着互联网服务的进一步深入，实体和虚拟的边界也会开始模糊，点对点价值转移的需求会被凸显出来，因此在互联网上面的 Value Super Highway 和 Value Transfer Protocol 必然会出现，而比特币网络加速了这一历史进程。

1.2 为什么设计量子链

自从 2009 年比特币代码开源以来，社区里面出现了很多 Altcoin 和其他区块链项目，有意义的 Altcoin 项目成为了区块链技术的试验田（一些毫无意义的 Altcoin 除外），对区块链技术的发展和成熟有一定的借鉴意义（例如 NameCoin 等），除此之外还有一些从不同角度拓展区块链技术边界的项目，例如 ColorCoin 协议，NXTCoin，Ripple 和 Stellar，BitShare，Dash，Maidsafe，Factom 等。 之后，还有致力于成为通用智能合约平台和去中心化应用平台的 Ethereum 项目。无数的开发者和社区人员一起参与和见证了区块链技术的快速发展，但是区块链行业不论是从技术角度，还是行业应用角度都还面临着很多挑战。

区块链技术面临的主要问题：

1 缺乏新型的智能合约平台，目前现有的智能合约平台主要是基于 Proof of Work（POW）
2，而 Proof of Work（POW）的共识机制很难被行业应用大规模部署。
3 不同区块链技术之间的兼容性，比如基于 UTXO 模型的比特币生态和基于 Account 模型的以太坊生态很难有兼容性。
4 共识机制本身缺乏灵活性，因为参与者的不同，在公有链中和联盟链中，对共识机制的要求是不一样的。
5 缺乏对行业合规性的考虑，例如在金融行业要求的 identity 和 KYC 部分，在现有的区块链系统中，很难保证。
6 现有区块链系统具备很大的封闭性，目前大多数的智能合约的触发条件大多来自于区块链系统本身，很少有来着外界的触发条件，缺乏与现实世界的交互。

针对当前区块链行业的挑战，量子链在区块链技术和理念上进行了一系列的创新：包括基于 UTXO 的智能合约模型，面向公有链和联盟链的灵活的共识机制，区块链主控合约的理念和实现，交易账本和智能合约账本的分离，Oracle 和 Data Feed 的设计和实现等，使得**量子链成为区块链世界与现实商业世界的桥梁。**

对比互联网技术的发展路径，我们发现不论是区块链技术本身，还是基于区块链技术的应用，都处于行业发展早期，有很多值得探索的方向。

因此我们希望可以构建一个全新的区块链生态系统，作为未来世界可选的互联网价值传输协议的可选项，并把整个区块链行业的**易用性**向前推进一步，这也是我们设计量子链的原因。

量子链致力于拓展区块链技术的**应用边界和技术边界**，使**普通互联网用户**能感受到区块链技术的价值，并构建一个全新的基于区块链技术的开发者和用户的生态系统。

1.3 量子链的设计原则

1.3.1 量子链兼容性设计

- **与比特币网络和以太坊网络的兼容性（Compatible with Bitcoin and Ethereum network）**

  比特币网络的生态系统是目前最大的一个区块链技术的生态系统，根据网络效应和马太效应（Matthew Effect）的影响，我们有理由相信比特币生态系统会进一步扩大和完善，也意味着更高的代码成熟度和更多的开发者。量子链在设计之初，就尽量保持与比特币系统的兼容性，例如同样使用 UTXO 的交易模型和相同的交易数据结构等，这也为以后借力比特币的 BIP 协议，提供了技术上的可能性。因此量子链后面可以兼容大部分的 BIP 协议，例如 闪电网络（lighting network）和侧链（sidechain）和驱动链技术(drivechain)和基于零知识证明的 Zcash 协议等。

  Ethereum 第一次将智能合约的概念从理论变成实际，从而拓展了区块链技术的边界，其中的 Ethereum Virtual Machine （EVM）虽然有很多值得改进的地方（例如：Transaction-ordering Dependence attack/Timestamp Dependence Attack / Mishandled Exceptions 等/ ），但是 EVM 是目前为止唯一 一个经过测试的智能合约虚拟机，因此保持和 EVM 的兼容性，就显得非常重要，因此量子链的虚拟机将保持和 EVM 的兼容性，所有在以太坊平台上面开发的智能合约，也可以在量子链平台上面运行。

- **量子链本身的向下兼容性（Downward Compatibility）**

  软件的向下兼容性也是一个非常重要的问题，使用旧版本创建的文件和智能合约，将能持续在新版本上面运行，而不用用户强制升级，这将会给用户带来很多便利。因为智能合约的特殊性和一次性部署，如果不能实现向下兼容性，将会给已经执行过的智能合约带来很大问题，也造成后期软件无法迭代和升级，也会出现 EVM2.0 和 EVM1.0 无法兼容的问题，这也是区块链系统软件设计者需要注意的问题。

1.3.2 量子链模块化设计思想

模块化的设计更利于软件的开发和维护，因此在 Qtum 中，我们分为以下 3 个大的模块：

- ✓ **量子链技术模块 Qtum Tech**：Qtum Core、Qtum VM、Qtum identity、Qtum Oracle and DataFeed、Qtum Storage 等；
- ✓ **量子链用户交互模块 Qtum UI**：Qtum IDE、Qtum Mobile、Qtum Web 、Qtum Node 等；
- ✓ **量子链商业路径模块**：Qtum Business：Qtum Financial、Qtum legal and risk、Qtum Industry、Qtum Competitor 等。

1.3.3 量子链安全性策略

A. Qtum 基础平台所用技术的可靠性

量子链的第一阶段 致力于提供一个面向不同行业的高可用的兼容 UTXO 模型的 POS 机制的智能合约平台。UTXO 模型是比特币网络非常核心的一部分，其中的代码具备比较高的成熟度，兼容 UTXO 模型，不但可以吸纳比特币生态系统的其他开发者和现有开发工具，还可以具有比较好的安全性。

关于共识协议部分，除了中本聪在比特币网络中采用的 Proof of Work（包括 Sha256/Sha3/scrypt/X11/X13/ 等其他变种），经过比特币网络 8 年的大规模测试以外，在公链服务中，另外一个广泛运用和被测试过的共识协议就是 Proof of Stake（POS/Dpos/POS 后续演进协议等），并且在 POS 后续演进协议中（POS2.0/POS3.0）已经逐步消除了针对 POS 协议的各种潜在攻击（例如：币龄攻击、POS 节点全部离线，预先计算 Hash 值攻击等）。

因此在 Qtum 系统中，我们第一阶段采取的共识协议的基础为 POS 协议，并添加激励措施，估计节点在线，我们可以称之为 IPOS（Incentive Proof of Stake）。 目前在我们的 Qtum 测试网络中，所采取的依然是 POS3.0 的协议，后续的开发中，会逐渐转移到 IPOS 协议。 关于共识协议的长期演进，为了取得更高的可拓展性，以及未来支持基于 Qtum 系统的私链服务，在共识协议的第二阶段，我们将采用 基于时间序列（POT：Proof of Time）和 Raft 协议融合的共识协议，期望取得金融级别的数据处理能力。在我们的第二阶段的共识协议的模拟中，设计目标为：区块时间 250 毫秒，确认时间：750 毫秒-3 秒。因为融合 Raft 协议，更适合在有限的网络节点中，随机选取 Leader 来进行周期性的记账。第二阶段的共识协议的设计，更多是为了以后基于 Qtum 系统的 私链网络或者联盟链网络而设计，是否可以用到 Qtum 的后续共识协议演进中，我们会做进一步探索。

关于智能合约的虚拟机部分，Qtum 目前测试网络中支持 EVM，EVM 也是目前为止唯一一个被测试过的智能合约虚拟机，后期重点包括 1） 开发一种更严格的智能合约编程语言；2）支持 EVM2.0（Wasm）；3）探讨在其他虚拟机平台上（LLVM、Lua、NodeJS）移植的可能性。

关于 Identity 和 Oracle（Data Feed）部分，详见后续章节。

B. Qtum 平台发行的安全性策略

Qtum 平台在发行前将会经过一系列严格测试，其中包括软件功能性测试、P2P 网络性能测试、潜在攻击向量测试、可靠性测试，安全审计和代码审核，Alpha 版本测试，Bata 版本测试，通过完善的软件测试流程，来控制软件质量。

1.3.4 量子链易用性策略

针对节点的易用性，我们将为用户钱包提供两种或者三种不同操作模式，包括 Simple Mode、Professional Mode、Expert Mode， 分别对应不同的用户操作系统和开发需求。

除此之外，我们将提供 Qtum 系统的完善 API 服务，将基于 Standard JSON-RPC 来提供远程的和本地的 API 调用服务，并考虑版本之间的兼容性。另外我们还将提供 Qtum 的 IDE 服务，可以开发和调试相关代码和服务。

此外，用户可以通过浏览器（Chrome or Firefox 等）访问 DAPP 服务，例如用户可以在浏览器中输入 **Qtum://DappName** 的形式，来访问 Qtum 系统中的去中心化应用程序，例如在 Qtum 系统的中一个去中心化拼车服务“车来了”，用户可以通过在浏览器中，输入：Qtum://chelaile，就可以访问相应的 DAPP 服务。

第二部分　量子链实现方案 Implementation

2.1　量子链公链 QtumChain

量子链公链致力于开发除了比特币和以太坊之外的新的生态系统，通过完善的设计，来实现和比特币长期技术演进和以太坊虚拟机相兼容的特性；并且以行业应用为导向，通过移动端 DAPP 开发策略，把区块链的技术优势带给不同行业的应用者和普通互联网用户。另外 Qtum 的公链系统注重智能合约的实际应用，将通过完善的 Oracle 和 Identity 部分的设计，给传统互联网企业（金融、物联网等）提供一个合规性的**开放的**区块链技术试验田。除此之外，Qtum 系统注重去中心化应用的开发，通过吸引第三方开发者加入，一起为普通用户提供移动端的去中心化应用，所有根据 Qtum 系统开发的第三方应用，Qtum 将通过完善的评价体系，给予开发者奖励。

Qtum 系统的初始设计目标是：**首个兼容 BIP 协议的 POS 智能合约平台，并通过 Identity 和 Oracle 和 Data feeds 的引入，在合规性上面，符合不同行业的监管需求。** 在 Qtum 平台上，我们将通过持续的区块链技术研发和创新，吸引第三方开发者、行业用户、普通用户一起共建 Qtum 平台和生态系统。

2.2 UTXO vs Account Model

2.2.1 UTXO 模型剖析

在比特币的网络中，UTXO（Unspent Transaction Output 未花费交易输出）是比特币交易的基本单位，通过交易的输入和输出，**比特币网络将金钱变化成一段数据结构**，区别于信用卡支付必须在加密安全网络中传输，比特币的数据可以在任何不一定安全的网络中传输（WiFi、蓝牙，NFC，表格等）。UTXO 可以是“一聪”（1×10^{-9} BTC） 的任意整数倍。尽管 UTXO 可以是小于 2100 万的任意数值，但是一旦 UTXO 被创造出来，只能作为一个整体被花掉。如果一笔交易需要的 BTC 小于某一个 UTXO 的值，那么该 UTXO 依然会被当做一个整体花费掉，并形成一个找零的 UTXO。

UTXO 可以看做被私钥的拥有者锁定的、并被整个比特币网络识别的比特币货币单位。

在 UTXO 模型中，被某一个交易消耗的 UTXO 被称为交易输入，由交易创建的 UTXO 被称为交易输出。通过这种方式，一定量的比特币在不同的私钥所有者之间转移，并在交易链条中不断消耗和创建新的 UTXO。一笔比特币交易通过所有者的私钥签名来解锁 UTXO，并通过使用新的所有者的比特币地址来锁定并创建 UTXO。在比特币网络的起始的阶段，矿工通过一种特殊的交易类型，Coinbase 交易创造的交易的输出（该交易没有输入），所产生的比特币，可以用于创建其他的 UTXO。

UTXO 被每一个全节点（Full Node）比特币客户端在一个储存于内存中的数据库所追踪，该数据库也被称为“UTXO 集”或者“UTXO 池”，新的交易构建时从 UTXO 池中消耗一个或多个输出，而比特币网络监测着以百万为单位的所有可用的 UTXO，世界上在比特币网络中并不存在“比特币余额”的概念，因为比特币网络上只会记录所有未花费的

UTXO，比特币的余额的概念更多是通过比特币钱包客户端派生出来的产物，比特币钱包通过扫描区块链并聚合所有属于该用户的 UTXO 来计算该用户的余额。

另外关于交易费用的问题，我们可以通过计算输入和输出的差额，来计算一笔交易的交易费用。

比特币网络交易费用= 交易输入总和—交易输出总和

比特币网络中的交易输入和输出数据结构：

表 2.1 比特币交易输入数据结构

比特币交易输入 数据结构		
字节数	字段意义	备注
32 bytes	交易	指向 UTXO 的指针
4 bytes	输出索引	即将被花费的 UTXO 索引
1-9 bytes	解锁脚本尺寸	解锁脚本字节数
变长 bytes	解锁脚本	解锁脚本
4 bytes	序列号	未使用的交易替换功能

表 2.2 比特币交易输出数据结构

比特币交易输入 数据结构		
字节数	字段意义	备注
8 bytes	比特币数量	单位为聪
1-9 bytes	锁定脚本大小	锁定脚本字节数
变长 bytes	锁定脚本	花费输出需要满足条件的脚本

由于每一个比特币的全客户端都会对每一笔交易按照一系列的规则，进行独立校验，一笔比特币交易所有的交易信息都包含在脚本中，如果任何一个节点按照脚本执行，并对结果的有效性进行了校验，那么其他所有节点必将得到一致性的校验结果，这也意味着一笔有效的交易对所有人都是有效的。

比特币网络中的每一笔交易的执行依赖于解锁脚本和锁定脚本。解锁脚本可以解决锁定脚本对某一输出值的阻碍，锁定脚本会在某一笔输出值上设置花费的条件。解锁脚本通常包含私钥的一个签名，也被称为 ScriptSig，锁定脚本通常会把一个交易输出锁定到一个比特币地址上（公钥的哈希 Hash 值）。

比特币全节点客户端会同时执行锁定脚本和解锁脚本来验证某一笔交易的合法性。客户端会先检索输入所指向的 UTXO，这个 UTXO 包含一个定义了花费条件的锁定脚本，然后客户端会读取试图花费这个 UTXO 的由客户端构造的输入中所包含的解锁脚本，并执行这两个脚本。 如果从解锁脚本处复制好堆栈数据之后，再执行锁定脚本的结果为真，那么说明解锁脚本有权使用该 UTXO，并发起新的交易。

一个简单的解锁脚本和锁定脚本的示意图如下：

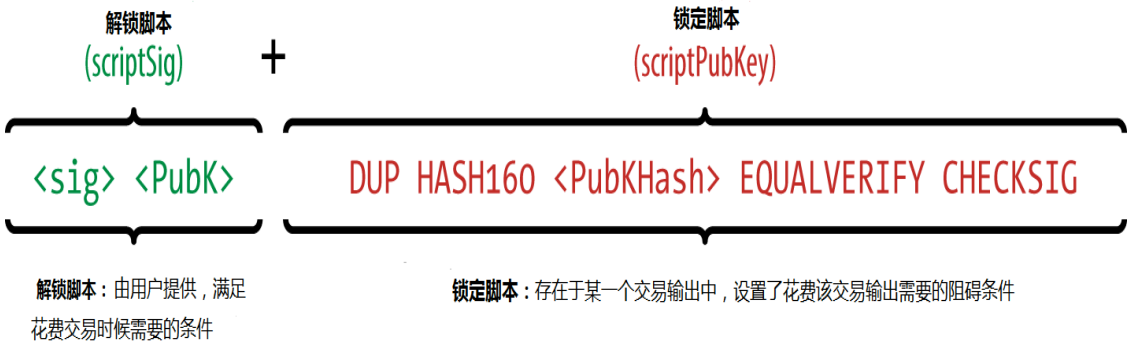

图 2.1 比特币解锁脚本和锁定脚本示意图

比特币在交易中使用脚本系统，与 FORTH（一种编译语言）一样，脚本是简单的、基于堆栈的、并且从左向右处理，它特意被设计成非图灵完备的，没有循环（LOOP）语句的一种系统。在比特币网络中，脚本系统对数据的操作都通过堆栈完成（主堆栈和副堆栈），堆栈是一个常用的抽象数据类型，最主要的特点就是后进先出（LIFO：Last In First Out）。

在比特币的客户端中，开发者把比特币客户端支持的脚本类型通过 Standard（）函数做了一个总结，在 Standard（）函数中包含 5 种类型的脚本：P2PKH（Pay to publickey hash）、P2PK（Pay to publickey）、MultiSignature（限 15 个私钥签名）、P2SH（Pay to Script hash）和 OP_Return。这 5 种标准的脚本类型实现了，通过公钥哈希支付、通过公钥支付、多重签名、通过脚本哈希支付、以及数据输出的功能。通过这 5 种标准的脚本类型，比特币客户端可以实现较复杂的支付逻辑。另外，一个非标准化的脚本类型也有可能被创建，但是必须找到一个愿意打包该非标准化的交易的矿工，该非标准化的脚本才会被执行。

我们以 P2PKH（Pay to publickey hash）为例，来说明脚本的产生和执行过程。假设我们需要向某一个面包店支付 0.01BTC 来购买面包，面包店的地址为：Bread Address。

则该交易的输出为：

OP_DUP OP_HASH160 <Bread Public Key Hash> OP_EQUAL OP_CHECKSIG

锁定脚本对应的解锁脚本为：

<Bread Signature> <Bread Public Key>

将两个脚本结合起来可以形成如下组合脚本：

<Bread Signature> <Bread Public Key> OP_DUP OP_HASH160
<Bread Public Key Hash> OP_EQUAL OP_CHECKSIG

只有当解锁版脚本与锁定版脚本的设定条件相符合的时候，执行组合脚本时才会显示结果为**真（Ture）**。要想执行组合脚本的结果为真，也就意味着，Bread Signature 的签名是有 Bread Address 对应的私钥所签名，是一个 Bread Address 的有效签名，只有这样交易执行结果才会通过（结果为真）。

虽然比特币的脚本语言包含很多的操作符，但是需要注意的是比特币脚本语言是非图灵完备的。在该脚本语言中，是没有循环功能的，这也意味着交易的复杂性有限，交易的可执行次数有限。脚本并不是一种通用的编程语言，这些限制也避免了潜在创造无限循环或者其他复杂逻辑漏洞的支付条件，从而对比特币网络的安全性留下隐患。

在 UTXO 模型中，我们可以通过公开的账本清晰的追溯每一笔交易的历史记录，并且可以做到完全透明，另外 UTXO 模型也带来了一定的并行处理能力，可以发起多地址到多地址的交易，为可拓展性带来一定借鉴。除此之外 UTXO 模型也带来了一定的隐私性，用户可以通过 Change address，来作为 UTXO 的输出地址。但是 UTXO 本身是无状态的，我们将通过一系列的创新的设计，实现基于 UTXO 类型的智能合约。

2.2.2 Account 模型剖析

与 UTXO 模型不同的是，以太坊是有账户体系的，在以太坊的白皮书中，我们可以看到以太坊的账户体系：

在以太坊系统中，状态是由被称为“账户”（每个账户由一个 20 字节的地址）的对象和在两个账户之间转移价值和信息的状态转换构成的。以太坊的账户包含四个部分：

- ✓ 随机数，用于确定每笔交易只能被处理一次的计数器
- ✓ 账户目前的以太币余额
- ✓ 账户的合约代码，如果有的话
- ✓ 账户的存储（默认为空）

以太币（Ether）是以太坊内部的主要加密燃料，用于支付交易费用。一般而言，以太坊有两种类型的账户：外部所有的账户（由私钥控制的）和合约账户（由合约代码控制）。外部所有的账户没有代码，人们可以通过创建和签名一笔交易从一个外部账户发送消息。每当合约账户收到一条消息，合约内部的代码就会被激活，允许它对内部存储进行读取和写入，和发送其它消息或者创建合约。

在以太坊系统中，通过一个有状态的账户系统来记录账户余额，每个账户余额的增加 / 减少更像现实世界中的银行记账方式，每产生一个新的区块，都会可能对全局状态造成影响。每个账户都有自己的余额、存储和代码区域。这样合约就可以调用账户或者地址，并且把相应的执行结果在存储区域进行存储。

在目前以太坊的账户系统中，通过 client/rpc，只能进行一对一的转账，也就意味着每次只能从一个账户转移到另外一个账户。尽管通过智能合约可以发送到更多的账户，但是这些内部交易只能在用户的账户余额上显示，却很难在以太坊的公开账本上追踪。

比特币网络的 UTXO 模型，保证了比特币交易的连续性和可追溯性，也是比特币架构的核心设计，考虑到比特币的网络效应和 UTXO 模型的优点，在 Qtum 的公链系统中，我们第一步决定采取基于 UTXO 模型。

2.3 共识机制 Consensus

量子链中的共识机制被设计成模块化的，可以类似插件一样实现插拔，可以适用公链和私链的不同应用场景。

在 Qtum 的共识机制的选取中，根据技术的可靠性原则和去中心化原则，我们最终选取 Proof of Stake 为基础的共识机制作为公链的基础共识机制。在基于 Qtum 的联盟链中，我们采用 Proof of Time 和 Raft 协议结合的共识机制，为行业客户提供服务。关于联盟链的共识机制的介绍，可以参考附件中 Qtum 的黄皮书（Yellow Paper）。

之前社区对共识机制的讨论较多，从 POW 到 POS 到 DPOS，再到 HyperLedger 提出的 BFT 共识机制。共识机制的本质在于在一个分布式系统中如何通过一些算法，最后取得数据的一致性。关于共识机制的讨论最后都会回归到计算机领域的分布式系统的一致性问题，之前这个领域已经有很多的研究和成果，例如分布式系统中的 FLP 定理和 CAP 定理指导人们如何根据具体的需求来设计共识机制。

在比特币的网络中，矿工通过比特币的全客户端一起参与到比特币网络的校验过程，通过工作量证明的方式，来随机碰撞 Hash 值，当矿工计算 Hash 值，满足一定条件时，我们就说该矿工挖到了一个区块。也即

$$\mathrm{Hash}(\mathrm{B\_Header}) \le \frac{\mathrm{M}}{\mathrm{D}}$$

Hash 函数代表 2 次的 SHA256 计算，取值范围是[0，M]，D 是[1，M]的一个整数，比特币网络的 SHA256 挖矿算法可以让每一个节点快速验证区块的有效性，并且 BlockHeader 每一个区块都随着 Nonce 和 extra Nonce 的不同而改变。整体挖矿的难度会根据网络的总算力而动态调整，根据共识协议，让网络有分叉产生的时候，我们会选取包含更多工作量的区块作为有效的区块。

后面根据挖矿算法的不同，还产生了其他的 Proof of Work 的算法，例如 Litecoin 的 Script 算法，Darkcoin 的 X11 算法，设计的初衷是抵制算力集中化，从而保证网络的去中心化。

目前社区所采用的 Proof of Stake 的代码大多分叉于 PeerCoin，而 PeerCoin 是基于非常老版本的比特币代码修改的，无法体现最近几年的 BIP 协议的特性和代码的完善性。

在 Qtum 的公链系统中，我们选取的共识协议的基础是 Proof of Stake，并会在最新的比特币代码的基础上，开发和部署 Proof of Stake 的共识协议。

在传统的 Proof of Stake 中，一个新区块的产生需要满足以下特点：

$$\mathrm{ProofHash} < \mathrm{coins} \cdot \mathrm{age} \cdot \mathrm{target}$$

其中 Proofhash 由 Stake modifier、未花费输出、当前时间一起决定。在这个规则中，如果有一个恶意的攻击者，他可以积累足够大的 Coinage 来发起一次双花攻击。另外 coinage

带来的一个问题就是节点再得到 pos reward 之后间断性上线，而不是持续在线来维护网络的完全。因此在 PoS 协议的改进版本中，我们把 Coinage 去掉，激励更多的节点同时在线。

$$Proofhash < coins \cdot target$$

另外在 Proof of Stake 中，节点的启动时间也会影响网络的安全性，在 Qtum 的 POS 协议中，我们将移除节点最小启动时间的限制，使节点可以在较短的时间内上线并且参与到网络的维护中。另外在 Qtum 系统中，为了鼓励更多的节点在线，我们会对于经常在线的节点设计更高的利息，即使你有较少的 coin，如果持续在线，也会有超过平均值的利息。

在 Qtum 公链系统中， 选取 Proof of Stake 的一些权衡： 去中心化的程度、 节点参与记账的难度、 网络的维护成本。

在 Qtum 联盟链系统中，涉及到身份识别和可信网络，更多的是在一个受限的小组中，周期性的随机选择记账节点问题，我们将会结合 Qtum 针对联盟链系统提出的 Proof of Time 和 Raft 协议，来设计相应的共识协议，具体的设计思路详见 Qtum Yellow Paper。**设计目标为：区块时间: 250 毫秒确认时间 750 毫秒- 3 秒，满足可拓展性和低延时的特点。**

2.4 合约和虚拟机 Contract and VM

在本小节，我们主要讨论合约，以及合约的执行环境虚拟机，以及量子链系统的提出的主控合约（Master Contract）的理念。

在本小节，这里的 Contract 我们指代 Blockchain Related Contract，并把区块链合约分成 Smart Contract 和 Master Contract.

- ✓ **智能合约 Smart Contract**：区块链合约代码通过虚拟机执行，并且不侧重链下数据的输入，借助于区块链网络本身提供合约触发条件，完成合约的执行。

- ✓ **主控合约 Master Contract**：区块合约代码通过虚拟机执行，侧重链下数据的输入（Oracle 和 DataFeeds），通过链下数据和区块链网络的共同输入作为触发条件，完成合约的执行。在 Qtum 的系统中，考虑到合约的实际用途，我们会设计完善的 Oracle 和 DataFeed 服务，把区块链的合约推向实际的商业应用场景。

我们来看一下比特币网络中的最简单的合约类型：多重签名。

在比特币的 2 of 3 的多重签名合约中，参与方同意把比特币资金放到一个交易的输入中，通过一个多重签名的交易，花费这币资金通常需要 至少 2 方的同意才能花费这笔资金。这种合约需要一个仲裁者（Mediator）作为第三方参与，因为多重签名合约会出现两种结果。第一种同意合约的执行结果，并且都发布自己的签名。第二种结果是其中一方不同意发布签名，造成这笔资金无法使用，这个时候就需要仲裁者参与，并且把资金给释放给他认为正确的一方。在这种合约过程中，我们可以把 Mediator 设计成自动提供可信数据源的 Oracle，并且从外部实际获取相应的数据，而不是从区块链网络本身获取合约执行的输入数据。

得益于以太坊的网络的合约执行环境 EVM，以太坊中的合约的编写和执行变得非常简单。但是现阶段以太坊上面的合约执行没有过多引入外面数据的干预，也就造成了在显示商业场景中的局限性。在 Qtum 的合约平台上，我们将把外部数据和干预措施抽象为 Oracle 和 DataFeed，期望通过主控合约形式，把区块链的合约带到具体的商业场景中。

我们来看一个简单的合约逻辑（QNT 为 Qtum 系统的 Token）：

```
function unlock(oracles, inputs) {
var better1Amount;
var better2Amount;
if (oracle.name.win == "barcelona") {
better1Amount = "1-QNT";
better2Amount = "0-QNT";
} else {
better1Amount = "0-QNT";
better2Amount = "1-QNT";
}
return {
"outputs" : [
{ recipient : "1MyuemkT4raRPPjhDwd9MyzbBNt9QAwKHc", value : better1Amount },
{ recipient : "1MdnpBCnFnjNFUaChHUMjfdW6bvhpR8WQi", value : better2Amount }, ],
"fee" : ["0.5-QNT", ]
}
}
```

如果在一场 Barcelona 对阵 Real Madrid 的比赛中，Barcelona 赢得了比赛（作为 Oracle 的输入），合约参与者 1 赢得比赛，否则输掉比赛。我们将在后面的章节中，具体探讨 Oracle 的创建和格式。

Ethereum 中的 DAO 事件暴露了智能合约设计中可能存在的潜在安全因素，因为在以太坊上面的智能合约一旦部署，EVM 就会通过预先定义的寻址指针和 OP_CODES，一步一步执行合约代码，并且根据合约的执行结果对全局状态进行转换。现实世界中的软件开发一般通过多次迭代来完成，但是智能合约一旦执行无法通过迭代改进，这个特性虽然符合区块链的准则，但是与现实世界的社会准则和商业准则有很大不同。因此在 Qtum 系统中，除了支持以太坊的智能合约，我们将建立现实世界与区块链的桥梁，通过链下规则的引入，把最简单的和最小集的合约需求写在区块链上面，例如合约的参与方和授权方等，并通过链下数据的输入来触发合约的执行结果，以及通过链下的仲裁来升级合约代码等。主控合约（Master Contract）的初衷在于引入社会规则和商业规则到区块链上面，使得区块链技术更容易对接现实世界的需求。

2.5 VM on 量子 UTXO 区块链（Qtum UTXO Blockchain）

2.5.1 Qtum 标准交易类型和合约交易类型

在 UTXO 模型的区块链系统中，VM 部分会独立于原有的脚本语言单独存在。在 Qtum 系统中，存在两种交易结构，分别是 Standard Transaction 和 VM Transaction。

- ✓ **Standard Transaction**：标准交易类型，我们标记为 V1 类型；
- ✓ **VM Transaction**：合约交易类型，我们标记为 V2 类型。

正常的类似比特币的交易类型我们标记为 Version1，Version1 的 transaction 保持与比特币网络标准交易的相似性，对于用到虚拟机的部分我们创造一个新的交易类型，我们标记为 Version2。

在 Version2 的交易类型中，我们会设计以下区域：

输入和输出 有一个“vm”区块，1 代表标准的交易类型，2 代表需要通过 VM 执行的交易类型。

这样新的 VM 合约可以通过创建一个新的输出来构建，并且随后可以通过一个标准交易来把 Qtum Token 发送到一个合约地址。其中 Token 可以通过新的操作符来分配，例如通过一个 assign-to 的 opcode 来分配。

2.5.2 Qtum 系统存储合约代码

通过拓展比特币网络的脚本语言，添加新的操作符，比如 OP_VM ，合约的 bytecode 将会被编码到交易输出中，合约字节数取决于系统设计的 MAX_SCRIPT_SIZE。

2.5.3 Qtum 系统中合约的创建

在合约创建的时候，通过创建人来建立合约

```
contract Escow {
 address owner;
 Escow() {
    owner = msg.sender;
 }
}
```

通过现有的脚本语言的解释器来执行 OP_VM 操作符，并把控制权转移给 VM 来执行相应的合约。

2.5.4 Qtum 合约花费量子币

合约作为某一笔交易的输出的时候，该输出又可以作为‘send’ opcode 的输入，Send opcode 可以把 V1 和 V2 类型的输出 和一定量的量子币发送到另外一个输出或者地址。

我们来看一个向某一个 Qtum 合约发送量子币的过程。

假设在第 100 个 block，我们有这样一个合约，合约的 TXID 为：1234，输出为 0

```
contract MoneySender{
   function sendmoney(outputScript){
      send(outputScript, this.balance / 2)
   }
}
```

有两笔 Qtum 的标准交易被发送到这个合约地址
value of 100 coins at block 150, tx id 1
assign-to 100.0
Value of 50 coins at block 200, tx id 2
assign-to 100.0

稍后我们在第 300 个区块的时候，调用该合约

call(contract100.0.sendmoney, myBitcoinAddressScript) //

最后我们将调用 Sendmoney 和 Send opcode. 并且该区块一笔有效的交易会包含所有的合约执行完成后的所有的 send opcode.这样在该区块的一笔交易中将会有 tx1 和 tx2 的 2 个输入，并且包含 myBitcoinAddressscript 和 change address 的两个输出。

2.5.5 Qtum 系统中合约状态的保存

每个合约脚本都有自己的状态，合约状态被存储在 Statedb 中，Statedb 可以通过区块链中个合约重建（reconstruction），我们把它叫做重建过程（reconstruction process）。状态总是可以重建/创建 re(constructed) 从现有区块中的合约。

其中 Statedb 应该可以重回的具体的交易中来处理一些冲突交易或者区块链中的短期的分叉。比特币网络所采用的 Berkley DB 并不是最优选择。

Statedb 的状态只会收到确认的区块的影响，未确认的交易和合约本身都不会影响 Statedb 的状态。

2.5.6 Qtum 系统中的密码学公钥方案

目前在以太坊中的公钥是通过预编译的合约来实现的，在 Qtum 系统中，我们将通过 VM 的 opcode 来实现。

2.5.7 Gas 机制的变化

EVM 系统中 Gas 的概念将会在 Qtum 系统中被移除，合约本身和合约调用合约都会自己来设定相应的费用，这个费用需要包含整个交易过程的费用（包括递归调用 recursive call）。为了防止 P2P spam，会对每一笔交易收取一笔小的费用。除了这个费用，记账节点和网络将会选择某一费率下他们可以执行的 opcode 的数量。

记账节点在处理交易的时候会遵守以下的流程：

1. 拒绝不包含最低费用的交易
2. 如果交易字节数太大，并且费用太低，这笔交易也不会被执行
3. 如果交易字节数很小，并且费用适中/偏低，记账节点会执行交易，但是如果对应的 opcode 太多，也可能中途被终止
4. 高的交易费用的交易会被优先执行，除非它们包含巨大数量的 opcode

在 Qtum 系统中，我们将根据实际的需求，会设计一个 opcode 的执行次数的上限，比如限制在 100k 以内的 operations。

2.5.8 合约账本（Contract Ledger）和合约的可读性（Contract Readability）

在 Qtum 系统中，除了基于 UTXO 模型的可追溯的 Transaction Ledger，我们还将构建一个合约内容的 **Contract Ledger**，方便大家的审计和阅读智能合约。

在以太坊系统中，智能合约的编写者，可以选择不发布合约明文内容和合约意图。在 Qtum 系统中，我们将构建一个 Contract Ledger 来存储所有的 Qtum 明文可读性强的合约内容，用户可以选着性的把自己感兴趣的合约代码和合约解释通过 P2P 的形式下载到自己的 Qtum 客户端。

Contract Ledger 的构建，可以给 Qtum 系统中的合约带来更多的透明性和可读性，以及可审计性。

2.5.9 Qtum 系统合约地址

在 Qtum 系统中，我们会根据以下规则构建新的合约地址类型，该地址区别于 Standard Transaction Address

Qtum Contract Address=
version + hash(spending_vout_txids[] + spending_vout_numbers[] + contract_bytes + contract_vout_number) + checksum;

其中的 Version number 的不同，后续可能对应不同的 VM 执行环境。比如 Version=1，代表兼容 EVM 的合约类型，Version=2 代表，兼容 Lua VM 的合约类型，这也给 Qtum 合约带来的后续的可拓展性。

2.6 Oracle 和 Data Feed

本小节论述 Qtum 系统中 Oracle 和 Data Feed 的理念和实现方法。

在 Qtum 系统中的区块链合约注重合约在商业社会中的实用性。Qtum 系统中的合约可能会需要和现实社会中的数据来做交互。比如 Qtum 系统中合约的执行有时候需要查询链外的数据（比如汇率、GDP、某个城市的温度、比赛结果等）。在 Qtum 系统中，Data Feed 代

表任何可以用来从链外取得数据，并把数据提供给区块链合约（或去中心化应用）的系统、程序、技术机制。

另外所有基于区块链的智能合约都需要所有节点执行相同的智能合约代码，也造成了智能合约会花费巨大计算资源，也变得异常昂贵。考虑到实际商业中用途，在 Qtum 系统中，针对联盟链，我们将创建一种链下合约的执行机制。正如我们在 2.4 小节中所描述的，我们把这种引入链下触发条件的区块链合约，叫做主控合约。除此之外，链下合约还可以给参与方更多的隐私保护，这一点更符合金融机构的需求。

另外，很多智能合约的应用（选举、彩票、cut and choose protocol）等都需要随机数的产生，我们会在本小节最后探讨随机数的产生方式。

2.6.1 通过 Oracle 实现链下合约的执行

在 Qtum 系统中，Oracle 代表可信的特定的机构、实体、节点、公钥地址。 Oracle 通过 data feed 的工作模式，来实现自己的职责。

关于区块链系统中最小信任的数据源之前有一些探索，从“Schellingcoin”到“Truthcoin”通过两阶段提高数据，并给接近平均值的数据源奖励，到通过参与者的权重来决定数据源的最终取值。也有一些项目尝试通过 Intel 最新的 Software Guard Extensions 来保证数据执行的可考虑，来最终提供数据源。因为 Oracle 的复杂性，在智能合约领域，目前还没有一个可以激励 Oracle 提供商的方式从而形成大规模的 Oracle 服务。

2.6.2 Qtum 系统中 DataFeed 实现思路

通常可信的数据源来源于被社会公认的机构，例如社会的消费者价格指数（CPI）和某一个重大比赛的结果。我们可以通过这些机构的网络服务来使用相应 API 接口，获取相应数据。比如通过 HTTP 的请求。

在 Qtum 系统中的 Data Feed，我们将引入博弈论的设计理念，通过不同数据源的 Deposit 来作为担保条件，并对诚实可信的数据源进行 reward。当有多个数据源的引入的时候，将通过对数据预先设置好的数据共识规则，来进行数据的处理。

Qtum 系统中 Oracle 的示意图：

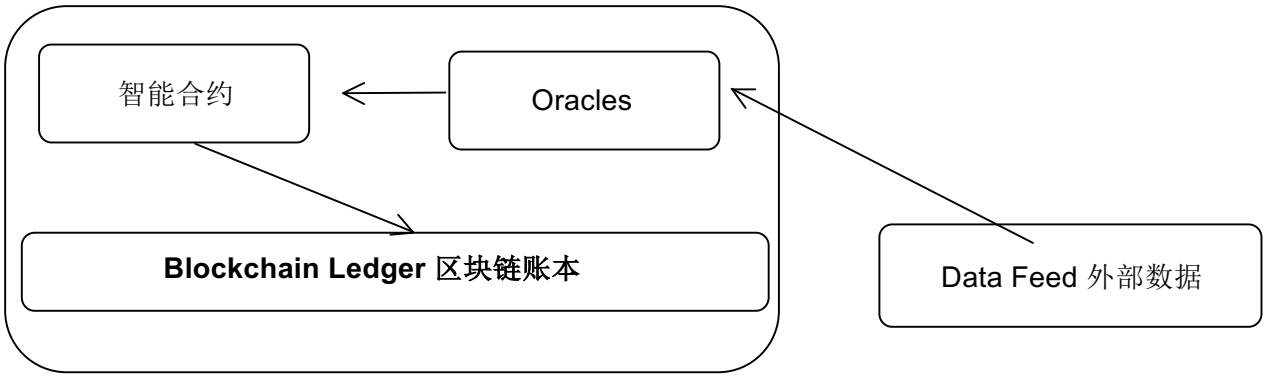

图 2.1 Qtum 系统中 Data Feed 和 Oracle 示意图

在 Qtum 系统中，Oracle data 可以是 Json file 的形式在创建和存储。

Oracle data file 的一个例子：

```
{
rate : {
"USD/EURO" : "1.10351",
"USD/GBP" : "1.31930",
"USD/JPY" : "0.00954"
}
}
```

比如这是一个智能合约引用的利率的例子。在智能合约中可以被引用：

```
function unlock(oracles, inputs) {
if (oracles.currency.rate["USD/EURO"] > 1.10) {
// make decision #1
} else {
// make decision #2
}
}
```

考虑到安全因素，智能合约不会直接调用外部的数据，在智能合约读取数据之前，Oracle 需要把相应数据写入到公开账本中。

Oracle 的创建

我们可以通过 Oracle 的“创建交易”，来创建 Oracle，在 Qtum 系统中分为以下两个步骤

1. 通过 OracleCreat Transaction，发起一笔交易；
2. 把数据写入到公开账本。

Oracle 创建交易
{
"id" : "hash of transaction, excluding signatures",
"transactionType" : "oracleCreateTransaction",
"oracleName" : "nameOfOracle",
"oracleProviderKey" : "MyuemkT4raRPPjhDwd9MyzbBNt9QAwKHc",
"oracleId" : "oracle_MyuemkT4raRPPjhDwd9MyzbBNt9QAwKHc"
"creation-payment" : "transactionId",
"name-payment" : "transactionId",
"fee-collection-recipient" : "@oracleProviderAccount",
"dataHash" : "MyuemkT4raRPPjhDwd9MyzbBNt9QAwKHc",
}

在 Oracle 的创建交易中，我们会制定一些参数，包括 TransactionType 和 Oracle ID 和 DataHash 等等。

关于 Oracle 的命名和支付，可以参见开发文档。

2.6.3 随机数方案

公平的区块链合约需要良好的随机数，之前社区有考虑过使用比特币的每个区块的块头和区块头部的 Hash 值来构建随机数，但是当利益足够大的时候，矿工可以选择欺骗。例如当一个竞猜合约的结果取决于随机数的时候，并且如果竞猜合约的价值大于 12.5 个 Btc，那么矿工可以选择不报块，使得竞猜结果有利于自己。另外比特币网络的出块时间的不稳定性，也造成了构建随机数过程中的困难性。

其他的随机数方案包括两阶段公布随机数结果。比如在一个选举过程中，需要随机抽取审计员，第一阶段：竞选委员会可以随机选取一个号码，并公布相应的 Hash 值。第二阶段：他们公布相应的随机数，并把该区块的去块头和该随机数利用，并产生新的随机数。但是如果竞选委员会不公布随机数，就会造成结果的无效性。

在 RanDAO 中，可以通过两阶段的提交产生较好的随机数，并且引入的激励机制，激励随机源成为诚实节点。

通过利用承诺协议（Commitment Protocol）和多阶段提交和博弈机制的引入，我们可以产生较好的随机数，来保证合约的公平性。

2.7 Identity and Privacy

Qtum 系统将通过智能合约管理量子链平台的上的用户。Qtum 系统将提供可选的身份识别模块，Identity 是区块链系统可以对接金融系统的前提条件。

在 Qtum 系统中，我们将区分 Identity 客户和非 Identity 客户。

Qtum 系统开发者将开发基于相应的 Identity 智能合约代码，并把代码开源给第三方。通过第三方征信机构的引入，在 Qtum 系统中，通过 Identity 智能合约验证的客户将会拥有更多的优先级。

例如在 Qtum 系统中的面向金融服务的 DAPP 中，Identity 验证的客户，将获得更多的权限。

关于 Privacy，因为 Qtum 系统兼容 UTXO 模型，ZeroCoin 为基于 UTXO 模型的加密传输协议，目前 Zcash 已经处于公开测试版，我们将通过与 Zcash 协议的融合，给 Qtum 系统中的智能合约和交易提供更多的隐私保护。

第三部分 量子链应用 Applications

3.1 去中心化应用（DAPP）

Qtum 系统致力从技术层面全面支持去中心化应用，尤其是通过移动端策略的引入，将不同的 DAPP 想法产品化，使普通互联网用户可以真正感受到区块链技术带来的价值。

面向不同行业的 DAPP 应用，可以把区块链技术带给更多的用户和行业。例如去中心化的社交、去中心化的存储和去中心化的域名服务、去中心化的计算服务等，通过激励机制的引入，将更深层次利用共享经济的理念，改变现有的 APP 市场和商业模式。

区块链技术为搭建去中心化应用（Decentralized Applications）提供基础架构。 在量子链中，通过完善的 Qtum API 的设计和 Docker 的分发，简化开发者的准备工作，使开发者可以快速上手相应的开发工作。并将通过 Qtum 系统内部的 Token 激励开发者开发出高质量的 DAPP。

3.2 多个行业的支持（Industry Oriented ）

在 Qtum 系统中，通过不同共识机制的引入和监管的需求，可以为行业发展需求也提供支持。

例如 Qtum 系统中，提供的基于 Proof of Time 和 Raft 协议融合的共识机制，可以满足可信网络中，对区块链速度和容量的要求，通过基于区块链技术的主控合约和 Oracle 和 Data Feeds 的引入，也可以引入更多线下的因素。通过 Identity 和 Privacy 的设计，可以符合金融行业的监管需求。

在 Qtum 系统中，可以支持多个行业的应用需求：例如 金融业、物联网、供应链、社交和游戏、慈善、数字资产和股权等。另外基于 Qtum 的智能合约和主控合约，通过图灵完备的编程语言，可以实现更复杂商业逻辑的支持，并将支持更多的行业。

关于 Qtum 的应用和行业支持，可以参考进一步的 Qtum 应用白皮书。

3.3 移动端策略 （Go Mobile）

“Go Mobile”是量子链开发团队的一个时刻谨记的原则，面向移动端策略也是推动区块链技术落地的一个重要环节，在量子链的生态系统中，我们不仅全面支持并推动移动应用战略，而且我们将会与第三方开发者，一起为用户提供移动端的服务，包括：移动端钱包、移动端 Dapp 应用、移动端智能合约应用等服务。 我们也鼓励第三方的开发者，加入我们，一起推动区块链技术在中国的落地，开发出普通互联网用户可以使用的区块链移动端服务。

获取更多开发信息：

Email： info@qtum.org
Website： www.qtum.org

参考文献：

[1] https://en.bitcoin.it/wiki/Category:History
[2] https://panteracapital.com/wp-content/uploads/The-Final-Piece-of-the-Protocol-Puzzle.pdf
[3] https://github.com/bitcoinbook/bitcoinbook
[4] https://github.com/ethereum/wiki/wiki/White-Paper
[5] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2009, https://www.bitcoin.org/bitcoin.pdf
[7] N. Szabo, Smart contracts, 1994, http://szabo.best.vwh.net/smart.contracts.html
[8] N. Szabo, The idea of smart contracts, 1997, http://szabo.best.vwh.net/idea.html
[9] Bruce Schneier, Applied Cryptography (digital cash objectives are on pg. 123)
[10] Crypto and Eurocrypt conference proceedings, 1982-1994
[11] David Johnston et al., The General Theory of Decentralized Applications, Dapps, 2015, https://github.com/DavidJohnstonCEO/DecentralizedApplications
[12] Vitalik Buterin, Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform, 2013, http://ethereum.org/ethereum.html
[13] Paul Sztorc, Peer-to-Peer Oracle System and Prediction Marketplace, 2015, http://bitcoinhivemind.com/papers/truthcoin-whitepaper.pdf
[14] PriceFeed Smart Contract, 2016, http://feed.ether.camp/
[15] V. Costan and S. Devadas, Intel SGX Explained, 2016, https://eprint.iacr.org/2016/086.pdf
[16] E. Shi. Trusted Hardware: Life, the Composable Universe, and Everything. Talk at the DIMACS Workshop of Cryptography and Big Data, 2015
[17] Ahmed Kosba et al., Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts, 2016, https://www.weusecoins.com/assets/pdf/library/Hawk%20-%20The%20Blockchain%20Model%20of%20Cryptography%20and%20Privacy-Preserving%20Smart%20Contracts.pdf
[18] Iddo Bentov and Ranjit Kumaresan, How to Use Bitcoin to Design Fair Protocols, 2014, https://eprint.iacr.org/2014/129.pdf
[19] Marcin Andrychowicz et al., Secure Multiparty Computations on Bitcoin, 2013, https://eprint.iacr.org/2013/784.pdf
[20] Ranjit Kumaresan and Iddo Bentov, How to Use Bitcoin to Incentivize Correct Computations, 2014, https://people.csail.mit.edu/ranjit/papers/incentives.pdf
[21] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas, Fair and Robust Multi-party Computation Using a Global Transaction Ledger, 2016, http://link.springer.com/chapter/10.1007%2F978-3-662-49896-5_25
[22] Guy Zyskind, Oz Nathan, Alex Pentland, Enigma: Decentralized Computation Platform with Guaranteed Privacy, 2015, http://enigma.media.mit.edu/enigma_full.pdf
[23] Joseph Bonneau et al., On Bitcoin as a public randomness source, 2015, https://eprint.iacr.org/2015/1015.pdf
[24] Dennis Mckinnon et al., RANDAO, 2014, https://github.com/dennismckinnon/Ethereum-Contracts/tree/master/RANDAO
[25] Dennis Mckinnon et al., RANDAO, 2015, https://github.com/randao/randao/blob/master/README.en.md
[26] Marcin Andrychowicz et al., Secure multiparty computations on Bitcoin, 2014, https://eprint.iacr.org/2013/784.pdf
[27] Arvind Narayanan et al., Bitcoin and Cryptocurrency Technologies, 2016, http://www.the-blockchain.com/docs/Princeton%20Bitcoin%20and%20Cryptocurrency%20Technologies%20Course.pdf

[28] Matt Springer, Is Bitcoin Currently Experiencing a Selfish Miner Attack?, 2014, http://scienceblogs.com/builtonfacts/2014/01/11/is-bitcoin-currently-experiencing-a-selfish-miner-attack/
[29] Edward Felten, Game Theory and Bitcoin, 2013, https://freedom-to-tinker.com/blog/felten/game-theory-and-bitcoin/
[30] Litecoin, 2011, https://litecoin.info/
[31] Evan Duffield and Kyle Hagan, Darkcoin: Peer-To-Peer Crypto-currency with anonymous Blockchain Transactions and Improved Proof-of-Work System, 2014, https://www.dash.org/wp-content/uploads/2014/09/DarkcoinWhitepaper.pdf
[32] Evan Duffield and Daniel Diaz, Dash: A Privacy Centric Crypto Currency, 2015, https://www.dash.org/wp-content/uploads/2015/04/Dash-WhitepaperV1.pdf
[33] Sunny King and Scott Nadal, PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, 2012, http://web.archive.org/web/20131228174819/http://peercoin.net/bin/peercoin-paper.pdf
[34] Novacoin, 2013, http://coinwiki.info/en/Novacoin
[35] Nxt, 2013, http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt

版本变更记录

1 量子链白皮书　帅初　V0.1 版本　2016 年 8 月 1　日
2 量子链白皮书　帅初　V0.2 版本　2016 年 8 月 15 日
3 量子链白皮书　帅初　V0.3　版本　2016 年 8 月 16 日
4 量子链白皮书　帅初 V0.6 版本　2016 年 9 月 18 日