Artificial Intelligence Blockchain (AIC) WHITE PAPER

Value Transfer Protocol and AI Platform and DAPP Platform

人工智能链白皮书

价值传输协议和人工智能应用平台和去中心化应用平台

基于区块链 **3.0** 协议

AIC 基金会：**aic@2048ai.com**

AIC 基金会

2018 年 02 月 12 日

摘要 Abstract

Artificial Intelligence Blockchain （简称**'AIC'**，）开发基于比特币和以太坊之上新的区块链生态系统，并致力于拓展区块链技术的**应用边界和技术边界**，使**全体人类**能享受区链技术和 AI 技术带来的划时代变革。

在 AIC 系统中，AIC 链将作为最底层的货币。我们将会为所有使用 AIC 的用户提供一个 AI 系统名为“AIS”(AI System)。每一个生成的 AIC 钱包地址的用户都可以免费申请领取，并为其训练技能，初始 AIS，无任何技能，需要用户自行训练。用户可以用 AIC 加速技能训练速度。如：语言识别技能，翻译技能，识别技能，财务技能等,企业用户可以直接和用户的 AIS 系统交互为用户提供服务。并将其应用至实际生活中。技能训练完成，可以交换技能。提供官方技能交易市场。技能为本地保存，官方不做任何保存。AIS 系统提供 OpenApi,可实现物理制造。第三方或用户可以通过对接 OpenApi 将其物理化。初期的 AI 需要用户引导和训练才具备一定的智能，中期的 AI 数据有一定的积累会逐渐的理解用户的需求，不管是工作或者生活都可以在一定程度上帮助用户，后期的 AI 会越发的智能，可能会具备一些人的特征，例如学习和模仿，会更明白用户的意图，为用户处理和规划生活，工作。

AIC 将采用 (Algorand BA★)股权证明。兼容区块链 3.0 全部协议（包括智能合约）。同时会打造一个 AIC.OS 的系统，任何的 VM 都可以在上面运行，理论上兼容所有的智能合约，并且会利用 Algorand 解决大矿池的隐患，51%算力的攻击，少分叉，扩展性，等目前区块链遇到的问题。我们会大力支持企业用户对接 AI 系统，会鼓励用户为自己的 AI 创造独特的技能，每一个 AI 都是独一无二的。

与此同时 AIC 将成为真正的市场化货币，而不是只能炒作的泡沫。至比特币诞生以来，一直被抨击是郁金香的泡沫，没有任何实际的应用。现在不一样了，我们要结合 AI 和区块链，让所有人都实现科技的飞跃，我们希望能在 AIS 系统上出现强人工智能，改变人类的历史。AI 将会为人类插上飞翔的翅膀

人工智能链白皮书
价值传输协议和人工智能应用平台和去中心化应用平台

周诚卓

第一部分　AIC 链实现方案 Implementation

1.1　Algorand 区块链协议

1.2.1 Algorand 的发展背景

Algorand 由 algorithm（算法）和 random（随机）两个词合成，顾名思义，就是基于随机算法的公共账本协议（public ledger）。Algorand 针对比特币区块链系统的几个核心缺陷进行了改进。

第一，比特币区块链系统的工作量证明共识机制需要消耗大量计算资源和能源。根据 Digiconomist 网站数据，截至 2018 年 1 月底，每产生一个比特币区块，需要运行 1.18*1022 次哈希运算。考虑到哈希函数作为随机预言（random oracle）的性质，比特币区块的产生过程，相当于掷一个有 1.18*1022 面的骰子，直到掷出某一特定的面为止。比特币区块链系统一年的耗电量，与秘鲁全国一年的耗电量相当，而且还在快速增长中。这些巨量计算和耗电，除了产生比特币区块以外，对人类社会几乎没有任何价值。

第二，比特币区块链系统要长期存续，要求 50%以上的计算资源掌握在诚实用户的手中。否则，恶意用户在力量占优时可能篡改区块链。但随着市场演变（中本聪应该没有预见到这种情况），比特币区块链系统中的计算资源集中在少数几个“矿池”中。这就构成了一个潜在的不稳定因素。“矿池”的存在也使比特币区块链系统偏离了其早期宣称的民主特征，形成了“矿工”和普通使用者这样不同阶层的使用者。从比特币历史上关于扩容的讨论以及多次分叉不难看出，比特币区块链系统已经形成了中心化程度很高的社区结构

第三，比特币区块链系统容易出现分叉。根据中本聪的白皮书[Nakomoto, Satoshi, 2009, “Bitcoin: A Peer-to-Peer Electronic Cash System”.]，当一笔交易被记入一个区块并接入区块链后，需要再等 6 个区块才能比较肯定这笔交易进入比特币的公共账本，而非在某一个分叉上。因为比特币区块链系统平均每 10 分钟才能产生一个区块，一笔交易从被记入区块到被确认需要 1 个小时左右时间。

第四，比特币区块链系统的可拓展性比较差。比如，一个比特币区块的大小为 1M，大约能容纳 2000 笔左右交易，因为平均每 10 分钟产生一个区块，比特币最高支持每秒钟 6 笔交易；相比而言，Paypal 平均每秒钟能支持 193 笔交易，Visa 平均每秒钟能支持 1667 笔交易[http://www.altcointoday.com/bitcoin-ethereum-vs-visa-paypal-transactions-per-second/]。从这个意义上讲，比特币是人类历史上投入产出比最低的社会和技术试验之一。对可拓展性问题，比特币闪电网络是目前很受关注的一个解决方案。

鉴于比特币区块链系统的上述缺陷，Algorand 的目标是：

1.能耗低，不管系统中有多用户，大约每 1500 名用户中只有 1 名会被系统挑中执行长达几秒钟的计算。

2.民主化，不会出现类似比特币区块链系统的“矿工”群体。

3.出现分叉的概率低于一兆分之一（即 10-18）。假设 Algorand 中平均每分钟产生一个

4.可拓展性好

1.2.2 Algorand 的基本设置

（一）用户和交易特征

Algorand 是一个公有链系统。用户（或者节点）加入 Algorand 不需要事先申请，可以随时加入。Algorand 对用户数量也没有任何限制。每个用户持有多个公钥。每个公钥均是一个电子签名机制的一部分，也就是有一个与之对应的私钥。每个公钥对应着一定数量的货币。每笔交易实际上是一个电子签名，该电子签名将一定数量的货币从某一个公钥转移给另一个公钥，并用前一个公钥对应的私钥进行加密。不难看出，Algorand 的这些设计，与比特币是一样的。

Algorand 要求系统中 2/3 的货币由诚实用户掌握。诚实用户的含义是：其行为遵守有关指引（主要指拜占庭共识协议，见下文），并且能完美地发送和接收消息。诚实用户以外的是恶意用户，恶意用户的行为可以任意偏离有关指引。

对恶意用户，Algorand 假设他们由一个“敌对者”（adversary）控制。“敌对者”能发起强大攻击，包括：

1.“敌对者”可以在任何时候瞬间地腐化任何他选中的用户，使其成为恶意用户（哪怕该用户之前是诚实用户）。

2.“敌对者”完全控制并且完美协调所有恶意用户。可以理解为，恶意用户的行为（包括发送和接收消息）完全由“敌对者”代理

3.“敌对者”控制所有信息发送，但必须满足一点：诚实用户发出的消息能在一定时间内（该时间只与信息的存储大小有关）抵达 95%的诚实用户。然而，“敌对者”几乎不可能伪造诚实用户的电子签名或者干涉诚实用户之间的通讯。

（二）区块链结构

在 Algorand 中，与交易有关的信息均存储在一系列区块中。每个区块主要包括：1.从上一个区块以来的交易信息；2.一个哈希指针，相当于对所有前序区块所含信息的一个摘要或浓缩；3.当期“领导者”电子签名后的“种子”参数（细节见后文）。这样，这一系列区块通过这些哈希指针连接在一起，构成了区块链。如果某一区块的信息被“敌对者”篡改，其与后序区块中保存的哈希指针就会出现不匹配。这就使得区块链难以被篡改。

Algorand 的上述设计与比特币区块链系统基本相同，但存在一个关键差别。目前，Algorand 是一个单纯的分布式账本协议，没有引入激励机制，没有发行类似比特币的数字加密货币。Algorand 中交易所用的货币是外生给定的（可以是任何法定货币或数字加密货币），交易只影响货币在不同用户之间的分配。而在比特币区块链系统中，“矿工”构建了被公共账本接受的区块后，就会得到系统给予的一定数量的比特币作为奖励。

（三）网络通讯

与比特币区块链系统一样，Algorand 假设用户之间的通讯采取“点对点传言”模式（peer-to-peer gossip）：当某一用户传播一条消息后，第一次收到这条消息的用户随机并且独立地选择他的一些“邻居”，并将消息传给“邻居”们。当没有用户是第一次收到这条消息时，这条消息的传播就终止了。

Algorand 对网络通讯的要求是：对任意大于 95%的可及性参数（reachability）ρ 和消息的存储大小参数 μ，总存在一个时间参数 λ（λ 只与 ρ 和 μ 有关），使得一个诚实用户发出的存储大小为 μ 的消息，在经过 λ 长时间后，至少被超过 ρ 的诚实用户接收。

（四）共识机制

Algorand 中，用户（不是全部用户，仅指被系统随机挑中作为“验证者”的用户，详见下文）通过一个拜占庭协议（由 Micali 教授开发，称为 BA★）对新区块达成共识。BA★执行起来非常快。大致言之，BA★每次循环有 3 个子步骤，在每次循环后均有 1/3 以上的概率能达成共识。一旦“验证者”对某一个新区块达成共识，超过一半的“验证者”再用自己的私钥对该区块进行电子签名（相当于认证），该区块就开始在 Algorand 网络中传播。

BA★的一个重要特征是：在点对点网络通讯下，BA★的参与者可更换（player-replaceable）。也就是，BA★每次循环的每一个子步骤均可由全新的、独立随机选择的参与者来执行。在这种情况下，BA★仍能正确、有效地达成共识。假设有上百万的用户，BA★每次循环的每一个子步骤的参与者可以完全不一样，而且每一批参与者都无法确定下一批参与者是谁，从而无法串谋。

（五）密码抽签（cryptographic sortition）

密码抽签是 Algorand 的关键创新，也是其得名的由来，其要点如下。首先，Algorand 创建并不断更新一个独立参数，称为“种子”。“种子”参数不仅不可能被“敌对者”预测，也不能被其操纵。

其次，在 BA★每次循环中，Algorand 基于当前 “种子”参数构建并公布一个随机算法（也被称为“可验证的随机函数”或 verifiable random functions，见下文）。该随机算法中的一个关键参数是用户的私钥，这个私钥只有用户本人才知道。

接着，每个用户使用自己的私钥运行系统公布的随机算法，得到自己的凭证（credential）。凭证值满足一定条件的用户就是这一轮的“验证者”（verifiers）。“验证者”组装一个新区块并连同自己的凭证一起对外发出。其中，在第一个子步骤中凭证值最小（按字典方面排序，见下）的那个“验证者”的地位比较特殊，称为“领导者”。

最后，所有“验证者”基于“领导者”组装的新区块运行拜占庭协议 BA★。在 BA★的每次循环中的每一个子步骤中，被选中的“验证者”都是不同的。这样能有效防止验证权力集中在某些用户手中，避免“敌对者”通过腐化这些用户来攻击区块链。

从上述过程可以看出:

第一，“验证者”在秘密情况下获知自己被选中，但他们只有公布凭证才能证明自己的“验证者”资格。尽管“敌对者”可以瞬间腐化身份公开的“验证者”，但已不能篡改或撤回他们已经对外发出的消息。

第二，所有“验证者”公布自己的凭证并进行比较后，才能确定谁是“领导者”，也就是“领导者”可以视为由公共选举产生。

第三，随机算法的性质决定了，事先很难判断谁将被选为“验证者”。因此，“验证者”的选择过程很难被操纵或预测。

第四，尽管“敌对者”有可能事先安插一些交易来影响当前公共账本，但因为“种子”参数的存在，他仍然不可能通过影响“验证者”（特别是其中的“领导者”）的选择来攻击 Algorand。接下来，对密码抽签和拜占庭协议 BA★做一个相对技术化的介绍。

1.2.3 Algorand 的密码抽签

密码抽签按两条线索执行：一是选出“验证者”和“领导者”；二是创建并不断完善“种子”参数。

（一）“验证者”和“领导者”的选择

假设在第 r 轮（可以理解为产生第 r 个区块的时间段）开始时，“种子”参数为 Qr-1（表现为一个由 0 和 1 组成的长度为 256 的字符串），PKr-k 是第 r-k 轮结束后系统中活跃的用户/公钥（活跃的标志是至少参与过一笔交易）的集合，其中 k 被称为回溯参数或安全参数。Qr-1 和 PKr-k 属于公共知识（common knowledge）。

凭证的定义。对每一个 PKr-k 中的用户 i，他使用自己的私钥对“种子”参数 Qr-1 进行电子签名后（用函数 SIGi 来表示）并输入哈希函数（用函数 H:来表示），得到自己的凭证 H(SIGi(r,1,Qr-1))(函数 SIGi 有多个输入参数时，表示将这些参数简单串联后再进行电子签名，下同）。哈希函数的性质决定了：

1.凭证是一个近乎随机的、由 0 和 1 组成的长度为 256 的字符串，并且不同用户的凭证几乎不可能相
同；

2.由凭证构建的 2 进制小数 0.H(SIGi(r,1,Qr-1))（也就是将凭证字符串写到小数点后）在 0 和 1 之间均匀分布。

一:“潜在领导者”的选择。对 0 和 1 之间的一个数，0.H(SIGi(r,1,Qr-1))≤p 发生的概率为 p，称所有满足此条件的用户为“潜在领导者”（也是这一步的“验证者”）。p 使得以 1-10-18 的概率，在所有“潜在领导者”中，至少有一个是诚实的。

二:“领导者”的选择。在所有“潜在领导者”中，存在一个用户 lr，其凭证按字典排序最小。也就是，对所有 0.H(SIGi(r,1,Qr-1))≤p 的用户 i，均有 H(SIGlr(r,1,Qr-1))≤H(SIGi(r,1,Qr-1))。用户 lr 就是第 r 轮的“领导者”。

三:“验证者”的选择。第 r 轮第 s 步（s>1）的“验证者”的产生程序与上文类似。在这一步中，每一个 PKr-k 中的用户 i 使用自己的私钥对“种子”参数 Qr-1 进行电子签名后并输入哈希函数，得到自己的凭证 H(SIGi(r,1,Qr-1))。对 0 和 1 之间的一个数 pr，满足 0.H(SIGi(r,1,Qr-1))≤pr 的用户就构成这一步的“验证者”

（二）“种子”参数的更新

用 Br 表示第轮结束后，拜占庭协议 BA★输出的区块。如果 Algorand 在第轮受到了“敌对者”攻击，Br 可能是空的，也就是不含有任何真实交易记录（用符号来表示）。比如，“敌对者”可能腐化了这一轮的“领导者”，使其将两个不同的候选区块发给诚实的“验证者”。考虑到这个情况，“种子”参数的更新过程是：

Qr=H(SIGlr(Qr-1，r)),如果 Br 不是空区

块=H(Qr-1，r）,如果 Br 是空区块

哈希函数的性质决定了，Qr 和 Qr-1 之间的关系近乎随机的。回溯参数或安全参数 k 则保证了，“敌对者”在第 r-k-1 轮几乎不可能预测到 Qr-1；否则，他将在第 r-k 轮引入恶意用户（也就是进入 PKr-k），从而能影响第 r 轮“领导者”和“验证者”的选择。

1.2.4 Algorand 的拜占庭协议 BA★

拜占庭协议 BA★相当于一个两阶段的投票机制。第一阶段，“验证者”对其收到的候选区块（为控制通讯成本，实际上用的是候选区块的哈希摘要）运行分级共识协议（graded consensus）,选出“验证者”共识最多的候选区块。第二阶段，“验证者”对第一阶段选出的候选区块，运行二元拜占庭协议（binary Byzantine Agreement），即要么接受它，要么接受空区块。需要强调的，在每一阶段中的每一个子步骤，Algorand 可能使用完全不同的“验证者”

以下为叙述方便，假设：

1.系统处在第 r 轮；
2.每一个子步骤都选出 n 名“验证者”，其中恶意“验证者”不超过 t，并且 n≥3t+1（也就是诚实“验证者”占比在 2/3 以上）。另外，引入计数函数#si(v)表示在第 s 步，“验证者”i 收到的消息 v 的次数（来自同一发送人的只能算 1 次）

（一）分级共识协议

步骤 1.1：运行密码抽签程序，选出“领导者”lr 和这一步的“验证者”。用 vi 表示“验证者”i 收到的并且他认识是来自“领导者”lr 的候选区块。

有 3 个原因使得 vir 不一定等于“领导者”lr 构建的候选区块：

第一，“验证者”i 辨认“领导者”的方法是从他收到的所有“验证者”凭证中找出按字典排序最小者。但因为网络通讯原因，“领导者”lr 发出的信息可能没有到达“验证者”i 处。

第二，“领导者”lr 可能被“敌对者”腐化，对不同“验证者”发出不同的候选区块。

第三，“验证者”i 本身可能是恶意的

步骤 1.2：“验证者”i 将 vi 发给其他用户（这对其他“验证者”都适用，下同）。
步骤 1.3：当且仅当“验证者”i 在步骤 1.2 中收到的某一个的 x 次数超过了 2t+1 次（即#2i(x) ≥2t+1），他将 x 发给其他用户。

输出：“验证者”i 按以下规则输出(vI,gi)：
如果存在 x 使得 #3i(x)≥2t+1 ， 则 vi=x,gi=2；如果 x 存在使得#3i(x)≥t+1，则 vi=x,gi=1；否则 vi=∅，gi=1，其中∅代表空区块。

Micali 教授证明了，分级共识协议的输出{(vi,gi):i=1,2,L……n}满足下列性质：对任意诚实“验证者”i 和 j，|gi-gj|≤1；对任意诚实“验证者”i 和 j，如果 gi,gj>0，那么必有 vi=vj；
如果存在 v 使得有 v'1=v'2=K……=v'n=v，那么对所有的诚实“验证者”i，均有 vi=v,gi=2。

因此，分级共识协议的输出{(vi,gi):i=1,2,L……n}存在两种可能的情形。
情形一：如果存在诚实“验证者”i，使得,gi=2，那么对任意其他“验证者”j，必有 gj≥1,vj=vi。此时所有诚实“验证者”输出的候选区块是一样的。当然，如果一开始的“验证者”收到的候选区块都是 v，那么最后一批“验证者”输出的也将都是 v。
情形二：对所有的诚实“验证者”i，gi≤1，并且他们输出的候选区块不一定相同。

（二）二元拜占庭协议

首先，基于分级共识协议的输出{(vi,gi):i=1,2,K……n}对每个诚实“验证者”赋值：如果 gi=2，那么 bi=0；否则 bi=1，。这些 bi 就是二元拜占庭协议的输入。对应着前面的讨论，此处也存在两种情形：

情形一：存在诚实“验证者”i，使得 bi=0。情形二：对所有诚实“验证者”i，均有 bi=1。

二元拜占庭协议可能经过多次循环，用计数器 ri 表示“验证者”i 经历的循环的次数。开始时，ri 赋值为 0。

步骤 2.1：“验证者”i 发出 bi

如果#1i(0)≥2t+1，那么“验证者”i 设定 bi=0，输出 outi=0，并停止执行协议（也可以认为他以后将一直发出 bi=0）；
如果#1i(1)≥2t+1，那么“验证者”i 设定 bi=1；否则，“验证者”i 设定 bi=0。

步骤 2.2：“验证者”i 发出 bi

如果#2i(1)≥2t+1，那么“验证者”i 设定 bi=1，输出 outi=1，并停止执行协议（也可以认为他以后将一直发出 bi=1）

如果#2i(0)≥2t+1，那么“验证者”i 设定

bi=0；否则，“验证者”i 设定 bi=1。

步骤 2.3：“验证者”i 发出 bi 和 SIGi(Qr-1,rj)。

如果#3i(0)≥2t+1，那么“验证者”i 设定 bi=0；

如果#3i(1)≥2t+1，那么“验证者”i 设定 bi=1；

否则，用 Si 表示所有给“验证者”i 发送消息的其他“验证者”集合，定义 $c = lsb\left(\min_{\gamma} H\left(SIG_{\gamma}\left(Q^{r-1}, \gamma\right)\right)\right)$（lsb 表示一个数的二进制表述中的最右边位置上的数字），“验证者”i 设定 bi=c，并且 ri 的计数往上调 1 位。鉴于哈希函数的性质，这实际上是将 bi 随机地、不被操纵地赋值为 0 或 1。

回到步骤 2.1，Micali 教授证明了，在二元拜占庭协议中，每个诚实“验证者”i 能以概率 1 停止并输出

outi∈{0,1}，并且：

1.（共识性）存在 outi∈{0,1}，使得对任意诚实“验证者”i，均有 outi=out；2.（一致性）如果存在 b∈{0,1}，使得对任意诚实“验证者”i，均有 bi=b，那么 out=b。

（三）拜占庭协议 BA★的输出

BA★的输出：对诚实“验证者”i，如果二元拜占庭协议的输出 outi=0，则输出 vi（即分级共识协议的输出）；否则，输出空区块Ø。

Micali 教授证明了，BA★满足拜占庭协议的要求：

1.（共识性）最后一批诚实“验证者”输出的区块是相同的；

2.（一致性）如果一开始的“验证者”收到的候选区块都是 v，那么 BA★的最终输出也是v

1.3 Algorand BA★ 共识算法详解

BA★是 PBFT 算法的改进。BA★ 算法分为三阶段：区块生成、GC 和 BBA★。算法的停止时间是不确定的，但大概率保证在有限步内结束。

协议里有两种角色：Leader 和 Verifier。Leader 在区块生成阶段创建区块；之后每一步里 Verifier 对区块进行共识。

符号

- (r,s)：第 r 轮第 s步
- ℓ^r：第 r 轮的 leader
- $SV^{r,s}$：(r,s) 的 verifier 集合。如果 s=1，则为 potential leader 集合
- $HSV^{r,s}$ 和 $MSV^{r,s}$：(r,s) 的诚实 Verifier 和恶意 Verifier 集合
- B_i^r：第 r 轮里节点 i 生成的区块
- B_ϵ：空区块。生成空区块的那一轮是不存在leader的。
- $sk_i^{r,s}$：节点 i 在 (r,s) 签名消息所用的临时密钥。每个 (r,s) 都有对应的临时密钥。
- $\sigma_i^{r,s}$：i 的签名 $SIG_i(r,s,Q^{r-1})$，用于证明 $i \in SV^{r,s}$
- $m_i^{r,s}$：节点 i 在 (r,s) 广播的消息。根据 s 不同，消息格式也不一样
  - $s=1$：$m_i^{r,1}=(B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$
  - $s=2,3$：$m_i^{r,s}=(ESIG_i(v_i), \sigma_i^{r,s})$
  - $s\geq 4$：$m_i^{r,s}=(ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$
- PK^r：在第 r 轮时已加入系统的所有节点的公钥集合

基本概念

种子

Q^r 是第 r 轮的种子，用于选举 Leader 和 Verifier。Q^r 的计算方式如下：

如果 B^{r-1} 是合法区块，则 $Q^r=H(SIG_{\ell^r}(Q^{r-1}),r)$ 如果 $B^{r-1}=B_\epsilon$，即空区块，则 $Q^r=H(Q^{r-1},r)$

Leader 选举

对于所有 $i\in PK^{r-k}$，计算 $.H(SIG_i(r,1,Q^{r-1}))\leq p$ 如果满足，则 i 为 potential leader。

其中 $H(\sigma_i^{r,1})$ 最小的节点为真正的 leader。

Verifier 选举

选举 Verifier 的方式和选举 Leader 的类似

对于所有 $i\in PK^{r-k}$，计算 $.H(SIG_i(r,s,Q^{r-1}))\leq p'$ 如果满足，则 i 为 Verifier

区块结构

区块结构不是协议重点，但还是提一下。 如果 B^r 是合法区块，则
$B^r = (r, PAY^r, SIG_{\ell^r}(Q^{r-1}), H(B^{r-1}))$ 如果 $B^r = B_\epsilon$，即空区块，则
$B^r = (r, \emptyset, Q^{r-1}, H(B^{r-1}))$

BA★共识

BA★由三个部分组成

1. 生成区块（s=1，即第一步）：所有节点检查自己是不是potential leader，如果是，则生成区块并广播
2. GC协议（2≤s≤3）：有点像PBFT的后两个阶段，verifier 会生成1个二进制值
3. BBA *（s≥4）：BBA共识的修改版。每次 BBA*都由3步组成，会不断地循环。什么时候结束是不确定的，依概率结束。

BA★将 leader 所生成的区块映射成二进制值，分别表示区块合法与否

- 合法：共识结束后生成一个正常区块
- 不合法：共识结束后生成空区块（ B_ϵ ）

约定

假设

1. 每一步中，都有 $|HSV^{r,s}| > 2MSV^{r,s}|$，即诚实Verifier比例大于Verifier集合的2/3。（P31.Parameters第1点）
2. 对于 $j \in HSV^{r,s'}$ 发的消息，最多经过 λ 时间能被诚实节点收到。（P45第6行）

执行前提

$s = 1$：节点 i 根据「Leader选举规则」检查自己是不是 potential leader。如果不是，结束该 step，否则执行相应规则。

$s \geq 2$：节点 i 根据「Verifier选举规则」检查自己是不是 vefier。如果不是，结束该 step，否则执行相应规则。

下文协议细节描述里，默认有这些检查。

签名

$esig_i$ 和 ESG_i 都表示用当前 (r, s) 的临时密钥来签名消息。

Step 的开始和结束

开始时间：一个节点在达成共识 B^{r-1} 后，会**同时**进入每个Step（论文 P44.Lemma 5.5(a)）。但并不是所有节点同时开始。

- 设第一个诚实节点达成共识 B^{r-1} 的时间是 T^r，则在 $[T^r, T^r + \lambda]$ 时间内，所有的诚实节点都会达成共识 B^{r-1}。
- 为什么是 $[T^r, T^r + \lambda]$，见分析的 2.1.a 和 2.1.b（论文 p50-52）

结束时间：Step 结束的条件有两种

- 达成 Ending Condition 条件
- 耗尽等待时间：第s步的等待时间为 $t_s = 2s\lambda - 3 + \Lambda$，$\lambda$ 表示 $m^{r,s}$ 的广播时间上限，λ 表示 B^r 的广播时间上限。【但为什么是 2λ ?】
  - $s = 1$ 时没有等待时间，因为不需要接收消息。
  - 论文中默认等待 t_s 后，能收到所有诚实节点在小于 s 的step里发送的消息。

Step 1 生成区块

这一步生成区块并广播

1. 生成区块 B_i^r
2. 生成 $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$。其中 $esig_i(H(B_i^r))$ 表示用 $sk_i^{r,1}$ 进行签名的，签名完后销毁 $sk_i^{r,1}$ 。
3. 广播 $m_i^{r,1}$ 和 B_i^r

备注

- 其他节点通过 $\sigma_i^{r,s}$ 验证是否有 $i \in SV^{r,1}$ ，以检查 i 是否有资格进行该步。
- 敌手收到所有的 $m_i^{r,1}$ 后，就能知道谁是leader，并立即控制它广播新的 $m_i^{r,1}$ ，阻止它原来的 $m_i^{r,1}$ 广播出去。这样敌手就可以控制每一轮的 leader。广播 $m_i^{r,1}$ 后销毁 $sk_i^{r,1}$ 就是为了避免这种情况发生，因为即使敌手控制了leader，也无法让他发送新的 $m_i^{r,1}$ 。

GC

Step 2：GC 第一步

1. 从收到的多个 $m_j^{r,1}$ 里找到 ℓ^r ，即 $H(\sigma_j^{r,1})$ 最小的节点。
2. 检查 ℓ^r 发来的区块 $B_{\ell^r}^r$ 的合法性：若合法，令 $v_i' = H(B_{\ell^r}^r)$；若不合法，令 $v_i' = \perp$ 。
3. 广播 $m_i^{r,2} = (ESIG_i(v_i'), \sigma_i^{r,2})$

备注

- v_i' 是 GC 协议要共识的值，在 BBA 协议中不会用到。
- 整个协议里只有这一步检查 Leader 和区块的合法性

Step 3：GC 第二步

1. 在收到的 $m_j^{r,2}$ 中，如果有超过 2/3 的 $(ESIG_j(v'), \sigma_j^{r,2})$ （他们的 v_j' 全部相同），则令 $m_i^{r,3} = (ESIG_i(v'), \sigma_i^{r,3})$；否则，则令 $m_i^{r,i} = (ESIG_i(\perp), \sigma_i^{r,3})$
2. 广播 $m_i^{r,3}$

Step 4：GC 输出

1. 收到的 $m_j^{r,3}$ 中，有 3 种可能出现的情况
    1. 如果有超过 $2/3$ 的 $(ESIG_j(v'), \sigma_j^{r,3})$，则令 $v_i = v', b_i = 0$
    2. 如果有超过 $1/3$ 的 $(ESIG_j(v'), \sigma_j^{r,3})$，且 $v' \neq \perp$，则令 $v_i = v', b_i = 1$ 。这里可以保证只有唯一的超过1/3的值。
    3. 否则令 $v_i = \perp, b_i = 1$
2. 广播 $m_i^{r,4} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,4})$

备注： $b_i = 0$ 表示区块合法， $b_i = 1$ 表示区块不合法

【思考】：在诚实节点中，似乎不会出现部分 $b_i = 1$ ，部分 $b_i = 0$ 的场景，而是会全部共识到合法区块 B_r 或空区块 B_ϵ 。恶意的 ℓ^r 和Verifier不能对下一步的诚实节点进行单播，因为他们不知道下一步的诚实节点是谁。

BBA★

在BBA*中，会不断对收到的历史 $m_j^{r,s'-1}$ 进行检查，查看是否触发Ending Condition

以下两个结束条件是互斥的

【Ending Condition 0】

- 条件：收到超过 $2n/3$ 的 $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$，且有 $s' - 2 \equiv 0 \bmod 3$；同时 v 在 $m_j^{r,1}$ 中对应的区块是合法的（注意 v 是区块的哈希）
- 满足条件则达成共识 $B^r = B_j^r$，将相应的 $m_j^{r,s'-1}$ 集合作为 $CERT^r$，停止该轮。

【Ending Condition 1】

- 条件：收到超过 $2n/3$ 的 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v), \sigma_j^{r,s'-1})$，且有 $s' - 2 \equiv 1 \bmod 3$
- 满足条件则达成共识 $B^r = B_\epsilon^r$，将相应的 $m_j^{r,s'-1}$ 集合作为 $CERT^r$，停止该轮。

Step 5：Coin-Fixed-To-0

当 $5 \le s \le m+2, s-2 \equiv 0 \bmod 3$ 时进行这一步时，如果触发Ending Condition 0或 1，则停止。否则等待 t_s 后

- 在收到的 $m_j^{r,s-1}$ 中，有超过2/3比例的 $m_j^{r,s-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$，则令 $b_i = 1$，否则另 $b_i = 0$
- 广播 $m_i^{r,s} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$
- 【为什么要令 $b_i = 1$ 和0】

Step 6：Coin-Fixed-To-1

和 Coin-Fixed-To-0 类似，当 $6 \le s \le m+2, s-2 \equiv 1 \bmod 3$ 时进行这一步时，如果触发 Ending Condition 0 或 1，则停止。否则等待 t_s 后

- 在收到的 $m_j^{r,s-1}$ 中，有超过2/3的 $m_j^{r,s-1} = (ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$，则令 $b_i = 0$，否则令 $b_i = 1$
- 广播 $m_i^{r,s} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$

Step 7：Coin-Genuinely-Flipped

当 $7 \le s \le m+2, s-2 \equiv 2 \bmod 3$ 时进行这一步，如果触发 Ending Condition 0 或 1，则停止。否则等待 t_s 后

- 可能出现三种互斥的情况：
  - 在收到的 $m_j^{r,s-1}$ 中，有超过2/3的 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s'-1})$，则令 $b_i = 1$
  - 在收到的 $m_j^{r,s-1}$ 中，有超过2/3的 $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s'-1})$，则令 $b_i = 0$
  - 否则令 $b_i = \mathrm{lsb}(\min_{j \in SV_i^{r,s-1}} H(\epsilon_j^{r,s-1}))$
- 广播 $m_i^{r,s} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$

Step m+3：最后一步

这一步比较特殊，不用检查自己是不是Verifier，所有节点都要参与。

如果触发Ending Condition 0 或 1，则停止。否则等待 t_s 后

- 令 $out_i = 1$ 和 $B^r = B^r_\epsilon$
- 广播 $m_i^{r,m+3} = (ESIG_i(out_i), ESIG_i(H(B^r)), \sigma_i^{r,m+3})$

达成共识 $B^r = B^r_\epsilon$，收集 $m_j^{r,m+3}$ 集合作为 $CERT^r$ 。

总结

【达成共识的情况】：有三种，分别是 Ending Condition 0、Ending Condition 1 和 Step m+3。

【Ending Condition】

1. 若因为收到超过 2n/3 的 $m_j^{r,s'-1}$ 而触发 Ending Condition，则在 $s < s'$ 的 step 里，肯定是因为耗尽等待时间 t_s 而结束step。
2. 触发 Ending Condition 0，则 s' 一定是 Coin-Fixed-To-0；如果触发 Ending Condition 1，则 s' 一定是 Coin-Fixed-To-1。
3. Ending Condition 要求收到的 m_j 个数要超过**确定值** $2n/3$，而其他的都只要求**收到的** m_j 中有2/3满足条件即可。

拜占庭共识要保证参与共识的诚实节点大于2/3，但随机选出的集合不能保证该条件。于是

1. 进行多次的随机选取（循环），只要有一次参与共识的诚实节点大于 2/3，就能达成共识。
2. 如果不进行多次随机选取（循环），则恶意节点每次把合法区块变成空区块的概率就会大大增加。

临时密钥

在每一个 (r, s) 里，节点 i 所用的使用临时密钥 $sk_i^{r,s}$ 签名消息。一旦消息广播出去，**立即销毁签名所用的私钥。**

这么做的理由：节点 i 发送消息 $m_i^{r,s}$ 的瞬间，敌手可以立即知道 $i \in SV_i^{r,s}$，就可以collud它，用它的 $sk_i^{r,s}$ 重新签名消息并广播。比如敌手可以这么攻击

- $s = 1$ 时：则每一轮敌手都可以控制 leader
- $s \neq 1$ 时，敌手可以有策略地控制verifier，使得每一轮都共识成空区块 B_ϵ

节点 i 每次会生成100万轮*180步的临时密钥（在加入系统或临时密钥用完时生成）。下面简单介绍两种生成方案（论文 p32）

第一种方案

1. i 生成 PMK 和 SMK
2. 广播 PMK
3. 通过 SMK 和 (r, i, s) 计算出100万轮*180步的 $sk_i^{r,s}$ 。计算完后销毁 SMK
4. 任何人可以通过 PMK 和 (r, i, s) 计算出 $pk_i^{r,s}$ （这一步论文里没有写，是我猜的）

第二种方案

- i 生成100万轮*180步的公私钥对
- 利用全部的私钥生成 Merkle Tree，广播 Root。
- i 广播 $m_i^{r,s}$ 时，附带公钥和 Merkle Tree的验证路径

1.4 独立虚拟机架构

AIC 会打造一个独立的虚拟沙盒系统（AIC.OS），从而稳定运行全部 VM，适配不同平台的 VM，并会之上
选取最合适的几款 VM，作为官方的适配，任何平台智能合约都可以运行在 AIC.OS 之上，用户不需要做出任何更改。会支持更多的语言，如:JAVA,Golang,Python,JavaScript，等具备图灵完备的语言。

关于 AIC.OS,将不在此做详细描述，会另出说明手册

1.5 VM on AIC Contract Blockchain

1.5.1 智能合约

智能合约是一套以数字形式定义的承诺，包括合约参与方可以在上面执行这些承诺的协议。区块链技术给我们带来了一个去中心化的，不可篡改的，高可靠性的系统，在这种环境下，智能合约才大有用武之地。智能合约是区块链最重要的特性之一，也是区块链能够被称为颠覆性技术的主要原因。

AIC 智能合约 包括以下特性：确定性、高性能、拓展性。其合约类型包括：验证合约、函数合约和应用合约。

从性能角度来说，AIC 采用了 AIC.OS 作为其智能合约的执行环境，能适配更好的 VM，其启动速度非常快，占用资源也很小，适合像智能合约这样短小的程序。通过 JIT（即时编译器）技术对热点智能合约进行静态编译和缓存可以显著提升。AIC.OS 的指令集中内建提供了一系列的密码学指令，以优化智能合约中用到密码学算法时的执行效率。此外，数据操作指令直接对数组及复杂数据结构提供支持。这些都会提升 AIC 智能合约的运行性能。

AIC 智能合约实现可拓展性的方法是通过高并发和动态分区的形式，结合其低耦合的设计完成的。低耦合合约程序在一个 VM 中执行，并通过交互服务层与外部通信。因此，对智能合约功能的绝大部分升级，都可以通过增加交互服务层的 API 来实现。

用什么语言编写智能合约？

从语言角度看 AIC 智能合约与以太坊的区别更为直观：与以太坊原创的 Solidity 语言不同，AIC 智能合约开发者可以直接使用几乎任何他们擅长的高级语言来进行 AIC 智能合约的开发工作。
AIC.OS 提供了这些语言的编译器和插件，用于将高级语言编译成虚拟机所支持的指令集。

第一批高级语言包括：
Java、Python

1.5.2 原子级跨链通信

可以预见，在未来较长的一段时间内，世界上将会有若干公有链以及成千上万的联盟链和私有链共存。这些独立的区块链系统都是价值与信息的孤岛，彼此之间无法互联互通。通过跨链互操作机制，可以将无数个孤立的区块链连接起来，使得不同的区块链上的价值可以相互交换，形成真正的价值互联网。

如果客户端不需要处理所有的交易会让多区块链间的整合更为轻松。 毕竟，一个交易所只需要关心交易所的入账和出账，别无他求。 如果交易所链条可以使用资金的轻量 merkle 证明，而不必非要完全依赖对它区块生产者的信任会是一个不错的主意。 至少一个链的区块生产者在与其他区块链同步时更乐意保持尽可能小的开销。
LCV 的目标能产生相对轻量存在性证明，使得任何追踪相对轻量数据集的人可以验证其有效性。在这种情况下，目的是为了证明一个特定的交易是包含在一个特定的区块中，区块包含在一个特定的区块链的已验证历史中。
比特币支持通过全节点的完整记录获取每年 4MB 大小的区块头信息来验证交易。 每秒 10 个交易，一个有效的证明需要 512 个字节。 这对于有 10 分钟间隔的区块链没有问题，但是对于 3 秒间隔区块链就显得不那么“轻量”了。

AIC.OS 软件使得任何一个人只要他拥有包含交易所对应区块之后的随意一个不可逆的区块头，他就可以进行轻量证明。 使用下面展示的哈希链结构就可以使用少于 1024 字节的大小来完成任意交易的存在性证明。 如果假设校验节点在过去几天内所有的区块头一直增长 (2MB 的数据)，那么验证这些交易将只需要 200 字节就够了。
将生产的区块与恰当的哈希链做关联使得开销增幅很小，这意味着没有理由不使用这种方式来生成区块。

当需要验证其他链时，有譬如 时间/ 空间/ 带宽 的多样化优化可以做。 追踪全部区块头 (420 MB/年) 将保持证明体积的轻巧。 只追踪最近的头可以提供最小长期存储和证明大小来获得。 另外，一个区块链可以使用懒惰的评估方法，即它记住过去证明的中间值哈希。 新证明只需要包含指向已知稀疏树的链接。 确切的方法将取决于那些包含对 Merkle 证明引用的交易所在的外部区块的比例。

一定密度的联系后，将变得更为高效，一个链会包含另一个链整个区块的历史和消除证据一起，这样就不需要通信便可以验证了。 出于性能原因，应最小化的跨链证明的频率。

当使用来自外部区块链的 Merkle 证明时，在已知所有交易均已验证和已知没有交易被跳过或遗忘之间有一个重要的差异。 虽然不可能证明所有最近的交易是已知的，但有没有间隙的交易历史是可以被证明的。 AIC.OS 在每个用户的每个传递的消息上分配了一个序列号。 一个用于可以使用
这些序列号来证明所有的消息由某个特定帐户处理，只需要看它是否是按序执行的。

1.5.3 整合以太访虚拟机(EVM)

以太坊虚拟机(EVM)是智能合约的运行环境。它是一个完全独立的沙盒，合约代码在 EVM 内部运行，对外是完全隔离的，甚至不同合约之间也只有有限的访问权限,AIC.OS 系统将会内嵌 EVM，用户在做智能合约迁移时，可以无缝进行。不需要做其他的操作。

1.5.4 令牌 Token 模型与资源使用

1.币数和交易记录 (硬盘)
2.CPU 与预计算 (中央处理器)
3.瞬时状态存储 (内存)

币数和交易记录有两部分，瞬时使用和长期使用。 一个区块链维持着所有消息的日志，这些日志最终由完全节点存储和下载。 通过消息日志可以重现所有应用的状态。

可计算债务是一个必须通过消息日志重新构建状态的计算结果。 如果可计算债务增长变得臃肿则有必要通过快照方式记录区块链状态，并丢弃区块链历史。 如果可计算债务增长过快，则它需要花费 6 个月时间来重放等值与 1 年的交易。 这很不可取，因此，可计算债务需要被细心的管理。

区块链状态存储是通过访问应用逻辑获取的信息。 它包括诸如挂单和账户余额等信息。 如果状态从未被应用读取则它不会被存储。 比如，博客发布的内容和评论如未被应用逻辑读取则他们就不应该存储在区块链状态中。 同时，发布的内容 / 评论的存在、投票的数量和其他属性要作为区块链状态的部分被存储下来。

区块生产者对外发布她们可用的带宽，计算能力和状态。

1.5.5 账户系统

在账户系统中，“账户”生成策略（每个账户由一个 24byte 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。账户包含四个部分

1.salt+随机数，用于确定每笔交易只能被处理一次的唯一标
示 2.账户目前的余额 3.账户的智能合约代码，如果有的话 4.
账户的存储地址 hash

AIC 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，用户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。

1.5.6 区块链和挖矿

第一，AIC 能在 1 分钟内确认交易，而且确认交易的时间随着用户数量的增加，变化不大

第二，将用户数固定在 5 万，测试不同区块大小对通量（throughput，可以用产生一个区块的平均耗时来衡量）的影响。可以看出，区块越大，构建区块的耗时越长，但对拜占庭协议 BA★的运行时间影响不大。

区块确认算法见(1.3 Algorand BA★ 共识算法详解)

1.5.7 关于费用

因为每个发布的到区块链的交易都占用了下载和验证的成本，需要有一个包括交易费的规范机制来防范滥发交易。比特币使用的默认方法是纯自愿的交易费用，依靠矿工担当监管者并设定动态的最低费用。因为这种方法是"基于市场的"，使得矿工和交易发送者能够按供需来决定价格，所以这种方法在比特币社区被很顺利地接受了。然而，这个逻辑的问题在于，交易处理并非一个市场；虽然根据直觉把交易处理解释成矿工给发送者提供的服务是很有吸引力的，但事实上一个矿工收录的交易是需要网络中每个节点处理的，所以交易处理中最大部分的成本是由第三方而不是决定是否收录交易的矿工承担的。所以这其实并不公平。

然而，当我们给出一个特殊的不够精确的假设时，这个基于市场的机制的漏洞很神奇地消除了。论证如下。假设

1.一个交易带来 k 步操作，提供奖励 kR给任何收录该交易的矿工，这里 R 由交易发布者设定，k 和 R 对于矿工都是事先（大致上）可见的。

2.每个节点处理每步操作的成本都是 C (即所有节点的效率一致)。

3.有 N 个挖矿节点，每个算力一致(即全网算力的1/N)。

4.没有不挖矿的节点。

当预期奖励大于成本时，矿工愿意挖矿。这样，因为矿工有 1/N 的机会处理下一个区块，所以预期的收益是 kR/N ，矿工的处理成本简单为 kC. 这样当 kR/N > kC， 即 R > NC 时。矿工愿意收录交易。注意 R 是由交易发送者提供的每步费用，是矿工从处理交易中获益的下限。 NC 是全网处理一个操作的成本。所以，矿工仅有动机去收录那些收益大于成本的交易。 然而，这些假设与实际情况有几点重要的偏离：

1.因为额外的验证时间延迟了块的广播因而增加了块成为废块的机会，处理交易的矿工比其它的验证节点付出了更高的成本。

2.不挖矿的节点是存在的。

3.实践中算力分布可能最后是极端不平均的。

4.以破坏网络为己任的投机者，敌对者确实存在，并且他们能够聪明地设置合同使得他们的成本比其它验证节点低得多。 上面第 1 点驱使矿工收录更少的交易，第 2 点增加了 NC; 因此这两点的影响至少部分互相抵消了. 第 3 点和第 4 点是主要问题；作为解决方案我们简单地建立了一个浮动的上限：没有区块能够包含比 BLK LIMIT FACTOR 倍长期指数移动平均值更多的操作数。具体地公式如下：

blk.oplimit = floor((blk.parent.oplimit * (EMAFACTOR - 1) + floor(parent.opcount * BLK LIMIT FACTOR)) /EMA FACTOR)

BLK LIMIT FACTOR 和 EMA FACTOR 是暂且被设为 65536 和 1.5 的常数

1.5.8 AIAPP 应用场景

AIS(AI System) 系统致力从技术层面全面支持去中心化结合 AI 的应用，尤其是通过区块链技术的引入，将很多的 AIAPP 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和 AI 结合带来的价值。

面向不同行业的 AIAPP 应用，可以把区块链技术和 AI 技术带给更多的用户和行业。
例如 AI 工作助手、AI 生活助手和 AI 学习助手、AI 出行助手、AI 管家等，TA 将带来的是便利的生活，快捷高效的工作，真正的解放人类的思维和双手的 AI，通过区块链的引入，将更深层次利用分布式和 AI 智能的理念，改变现有的 APP 市场和商业模式。并且进一步的将提供物理 API，可以让个人，企业，政体，等实体真实接触你自己的 AI。

区块链技术和智能 AI 为搭建去中心化应用（AI Decentralized Applications）提供基础架构。在 AIS 中，通过完备的技术文档 和 本地化 AI，简化用户的学习步骤，使用户可以快速上手相应的开发工作。并将通过 AIC 系统内部的 Token 激励用户训练出强人工智能

1.5.9 治理机制

链上治理：AIC 管理代币的持有人是 AIC 网络的所有者和管理者，通过在 AIC 网络上构造投票交易来实现管理权，通过获得 AIC 管理代币所对应的 GAS 燃料代币来实现 AIC 网络的使用权。 AIC 管理代币可以被转让。

链下治理：AI 理事会是 AIC 项目的创始人组织成立的常务管理机构，下设管理委员会、技术委员会和秘书处，分别负责战略决策、技术决策和具体执行。AI 理事会向 AIC 社区负责，以推广和发展 AIC 和 AIS 生态为首要工作目标。

1.6 其他事项

1.6.1 货币和发行

AIC 的分发：
发行策略是初始发行 2 亿，每年固定增发 30%，但会根据当年消耗情况增发上下浮动比例不超过 30%的 Token

AIC 初始 2 亿管理代币分为两部分：

第一部分 1 亿份 AIC 用于按众筹轮次和比例分发给 AIC 开发经费众筹的支持者，。

第二部分 1 亿份由 AI 理事会管理，用于支持 AIC 网络的长期开发、运维和生态发展。该部分的 AIC 处于锁定期，在 AIC 网络运行达 1 年时方可解锁被使用。这部分 AIC 不会进入交易所交易，仅用于长期支持 AIC 项目，计划按如下比例分配使用：

1. 2000 万份 用于激励 AIC 开发者和 AI 理事会成员
2. 2000 万份 用于激励 AIC 周边生态开发者
3. 2500 万份 用于交叉投资其他区块链项目，所获得代币归属于 AI 理事会，并仅用于 AIC 项目
4. 1500 万份 机动使用
5. 2000 万份 投入 AIS 的相关系统研发每年

使用的 AIC 原则上不得超过 3000 万份

1.6.2 扩展性

扩展性问题是所有区块链常被关注的地方，与比特币一样，区块链也遭受着每个交易都需要网络中的每个节点处理这一困境的折磨。比特币的当前区块链大小约为 20GB，以每小时 1MB 的速度增长。如果比特币网络处理 Visa 级的 2000tps 的交易，它将以每三秒 1MB 的速度增长（1GB 每小时，8TB 每年）。AIC 可能也会经历相似的甚至更糟的增长模式，因为在 AIC 区块链之上还有很多 AI 应用，而不是像比特币只是简单的货币，所以 AIC 承诺将时刻关注新的协议提出，适时的调整底层技术以便更好的服务于用户的交易和使用

第二部分 人工智能应用 AI Applications.

2.1 人工智能应用（AI APP）企业和用户

1.企业商用

在企业方面 AIS 系统会帮助在 AIC 上构建智能应用的企业，开放基于和用户 AI 对接的 API，企业开发的虚拟应用和硬件应用，只需要对接 AIS 预留的接口，即可和用户 AI 发生交互，进行数据交换，指令操作等功能，从而实现，商用开发的需求。举个例子：A 企业在 AIC 构建了一个基于 AIC 区块链的智能出现应用，用来帮助用户规划出行方面的需求，该应用可能包括，路径规划，地点选择，推荐景点，等一系列功能。但应用如果不对接用户自己的 AI，所属功能其实大多数用户看到的推荐结果，路径规划可能千篇一律。但如果接入用户的 AI，让用户的 AI 来使用这些功能，自主 AI 会帮助用户，选择他们最合适的时间，选择最适合的路径，最好的景点。真正让用户做到无需动手，就可以让一切都做到最好的。更甚者，AI 对接的 A 企业的驾车应用，A 企业只需要和 AI 对接好 API，用户的 AI 就可以自主驾驶，从规划出行到自动驾驶，全部一步到位。
初期 AI 的交互可能只是发生在虚拟应用上，但已经可以满足一部分的企业的商用需求，如京东推荐，游戏推荐，新闻推荐，自动理财，料理推荐，景点推荐等，将发生巨大的改变，从原来企业想破头的推荐算法，研究用户喜好等，变成用户 AI 自主抉择，选取用户所喜好的，因为用户 AI 就如同第二个 TA，企业只需要提供源源不断的信息，而用户 AI 会过滤掉所有用户不喜欢，不想要的信息，展示给用户的永远是最好的。更甚者用户 AI 甚至可以帮用户做决定。
很多目前的互联网企业商业模式将会发生巨大变化，因为 AI 可以更快的检索信息，AI 可以更好的给出实际的数据评判标准，AI 可以不断的改进，AI 可以随用户心意而改变。企业将不用浪费巨大资金去研究用户喜好，广告商也不用到处投放广告，因为再精准的算法，也不比不上用户的第二大脑。
而进一步的物理商用，硬件企业可以制造出硬件并预留接口能对接上用户 AI，而用户 AI 将可以直接操作硬件，但这需要很多技能的组合，并且 AI 需要有足够的计算力，才能实现。但物理 AI 可以在初期现先行应用在智能家居中，现在的智能家居很多都不智能，因为没有一个足够好的大脑来指挥它们，比如，智能音箱，智能门锁，智能电视，其实都不够智能。因为程序是定式的，无法随用户心意变换。而用户 AI 则可以指挥它们。举个例子，用户 AI 在用户回家时，可以遥控提前精准计算好车速，打开车库门，可以在用户距离家门前一秒前，打开家门，可以在用户疲惫时，提前打开加热淋浴，可以控制音箱放出符合用户心情的音乐，这些都是可以在初期就能实现的，并且不是很复杂的操作。在进一步的就是将用户 AI 制造出来，到这一步非常困难，也非常简单，因为需要操作那么多硬件子系统需要很多技能辅助，如果是拟人的机械，比如：步伐技能，手臂技能，面部技能，雷达探测技能，光学感知技能，等很多技能的组合，并搭配一个强大的计算系统才行，如果是非拟人的机械，相对会简单很多。
物理 API 和虚拟 API 都会相继开放出来，会有专门的 API 文档和强大的 AIS 系统做支持，以满足企业用户的需求。

2.用户使用

用户的 AI 运行和存储形态分三种，(云端运行,云端存储，CC)，(本地运行,本地存储,LL)，(本地运行,云端存储,LC) (云端运行,云端存储，CC):可以为用户省下很多的钱，不需要专门的手机或计算机来保存 AI 的数据，也不需要占用用户的计算资源，可以横向扩展 AI 性能，永远不会丢失数据，但官方不保证 AI 的纯正性,不保证 AI 的安全性。(本地运行,本地存储,LL):用户需要专门的计算机或手机来存储 AI 的数据，AI 运行起来的计算资源如果过大，用户的计算机比较差，会有卡顿的现象。纯正性，安全性用户自行控制。(本地运行,云端存储,LC):用户可以在云端存储 AI 相关的数据，可以随时读取，然后再本地运行 AI。

用户的 AI 前期需要一定的编程基础才能训练技能，中期需要了解编程基础，后期不需要编程基础。初期官方会提供详细的教程。前期会提供大量奖励，鼓励用户自主训练技能技能可以在训练成型后，进行交换。官方提供平台，官方会有免费的技能。
用户的 AI，独属于用户个人。
用户的 AI 可能因技能不同，会有高低强弱之分。

用户的 AI 在初期的会如同一张白纸，什么都不会。用户可以为其搭配技能。技能训练需要消耗的大量的计算资源，如让 TA 去打字，需要 TA 明白什么是键盘，什么是字母，什么是语言，等很多东西需要 AI 能判断出来，这些东西就和人一样需要去记忆和理解，AI 需要大量的反复运算和判错才能如人一样，而学习的事物越复杂，消耗的时间越多，计算资源也就越多，如果是用户用自己的计算机，可能需要上千次，好几星期才能让 AI 学会，为了节约用户的时间和为了让计算资源更加的集中，官方提供加速平台，而 AIC 就可以用来加速训练，技能学会后会固化下来，用户可以赠予给其他的 AI，也可以放在技能市场进行交易，但不要以为技能越多越好，如果有重复的技能，如浪费大量的用户的系统资源。比如一个打字技能 A，一个打字技能 B，差异可能不大，但 AI

在选择时可能会在使用完 A 后继续使用 B，这样两个操作都会被执行，反而浪费了 AI 的计算资源。初期的 AI 和婴儿一样，什么都需要教学，但 AI 的成长速度和计算能力，是人远远比不了的。中期的 AI 将会相对智能些，技能可以相互组合，并且经过 AIS 系统的不断的迭代，将会相对智能些，将会对用户的工作和生活带来改变，比如你是会计，你将不需要一个个把金额记录到系统里面，因为 AI 的识别能力，将会识别出纸面上的金额，自动读取计算机表格里面的金额，然后进行计算和归纳，但 AI 不会代替人来做这些工作，而是辅助用户。生活中，AI 将会提供用户喜好的商品，喜好的去处，喜好的新闻，因为中期的 AI 经过数据的积淀，已经能为用户着想了。后期的 AI 将会具备，反馈和模仿能力等，人所具备的能力。会更好的服务用户并会带来对日常生活，工作巨大的改变。AI 会成长成什么，由用户决定

2.2 加速平台（Accelerate the platform）

用户的 AI 要想快速掌握某项技能，就需要消耗资源和时间，如果是用户自行训练，第一，没有足够的资源。第二，没有 AIS 系统做支撑，必然事倍功半。第三，官方不会保存任何技能。
如果是其他企业或者用户推出的加速平台，即使能做到第一点，第二，第三，无法保证。并且官方需要 AIC 作为消耗品。如果后期需要，官方会开放加速平台源码，让其他的企业或个人可以自行组建加速平台，但全部需要使用 AIC，否则一律不许自行组建。
加速平台，会根据使用的 AIC 数量，加大算力的投入，并给出预估完成的时间，但最低不会低于 1 小时。数据来源可以由用户提供，也可以使用官方提供的，但这个数据来源，只会再初期和中期有，后期，预估是不会需要任何数据来源，AI 会自行搜索和收集数据。加速平台用户自主选择加速模式，加速 GPU，训练次数等，训练完成后，会同步给 AI，官方会自动删除相关训练代码和技能和数据。官方只提供加速，不对技能是否符合要求
做任何保证。训练过程中可随时中断，训练完会通知用户。

2.3 技能市场（Skills market）

用户训练完成的技能可以放在交易市场进行交易，官方鼓励用户自主训练技能，但技能如果有危害，官方不会允许上架。技能交易在前期不会收取任何的手续费，后期视情况而定。技能可以被重复购买，技能可以公开售卖，也可以不暴露用户 ID 进行售卖，技能需要简要的说明和演示，技能购买完成后不可退换。交易全部以 AIC 进
行。价格用户自定，但不允许随意定价，压价。破坏市场秩序。

2.4 API 开放（Open Api）

用户的 AI 都会预留 api，用户给个人开发者或者企业用户使用，会出详细的文档说明，初期开放的只会有线上虚拟 api，用以读取 AI 的数据，或者让 AI 去操作应用的 api。这样就可以打通很多系统之间的交互，AI 可以直接操作应用的界面和功能，省去用户去操作的繁琐的步骤，也可以读取 AI 的数据，从而获得用户的信息，能获得什么样的信息，用户自行决定。企业可以通过获得的信息来决定给用户展示什么数据，或者开放数据接口给
AI,让 AI 筛选数据。在虚拟 api 成熟后，官方会逐步的开放物理 api 的实验。

第三部分 工作计划 Plan Work

3.1 众筹计划（Crowdfunding plan）

众筹时间:2018-03-10 13:00 至 2018-03-31 13:00
众筹币数: 1 亿
众筹硬顶：5000 万美元
支持币种：ETH
兑换比例：1 ETH=100 AIC

不设置最低成功限制，筹满或 ICO 时间到期为止

3.2 激励计划（Incentive plan）

激励池有 1 亿的 AIC，将会被用于创始团队的激励，重点会用于 AIS 系统的开发，如加速平台的开发，AI 的开发，激励开发者用户。等方面，目的只有一个，就是为了让所有用户和参与者都能实现个人价值

3.3 开发计划(Development Plan)

1.开发第一版，基于 3.0 的协议的 AIC 和简单版本 AI 预计 1 个月 2018 年春
2.开发第二版，完善 3.0 基础协议，开发官方简单技能图谱，升级 AI 性能 预计 3 个月 2018年夏
3.开发第三版，开放技能市场，开放简单加速平台 预计 3 个月 2018 年秋
4.开发第四版，升级加速平台，简化技能开发流程，支持各个平台深度学习 预计 3 个月 2018 年冬

5.开发第五版 优化 AIC，优化 AI，优化技能市场，优化加速平台 预计 6 个月 2019 年春
6.开发第六版 开发物理 API 实验版本，支持 http 协议到 tcp，mqtt，等一系列协议预计6个月 2019 年秋
7.开发第七版 专注 AI 升级优化，打造全功能性 AI 平台 2020 年春

(ps:请随时关注 Github 获取开发具体情况)

3.4 项目预算 (Project budget)

预计初期募集的 5000 千万美元，将三分之一用于 AIC 系统的研发，三分之二 AIS 系统的开发，官方会定期公开人员情况和研发进度。以保证信息的透明度

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] https://github.com/EOSIO/Documentation/blob/master/zh-CN/TechnicalWhitePaper.md
[29] https://github.com/ethereum/wiki/wiki
[30] https://arxiv.org/abs/1607.01341
[31]https://zhuanlan.zhihu.com/p/32282367