Algorand Spec翻译

0、原文

- https://github.com/algorandfoundation/specs/blob/5615adc36bad610c7f165fa2967f4ecfa75125f0/overview/Algorand_v1_spec-2.pdf

1、区块链和余额表

- Algorand区块链是一组区块的有序链表。最开始，它由单独一个区块B^0组成，称这个区块为创世区块。新的区块会逐渐添加到区块链中。一旦区块B^r被确定，共识协议就开始执行生成并确认下一个区块B^r+1的过程。我们可以认为共识协议的第r轮执行过程就是生成并确认区块B^r的过程。第0轮代表区块链的初始化，小于0的轮数都代表第0轮，例如，B^−1=B^0。
- 区块B^r包含了一组交易集和一个随机种子Q^r，以及一些辅助数据用来验证数据的合法性。
- Alogrand区块链维护了一组用户的账号数据集。每个账号数据对应一个唯一用户，与用户是一一对应的。
- 账号由一个32字节的公钥来标识。这个公钥由Ed25519算法生成或者由多重签名账号生成。
- 在任意一轮，用户有多个状态，可以在线或者不在线。在线的用户参与共识协议决定下一个区块；不在线的用户则不能这么做（但他们可以发送交易）。而且在线的用户必须维护一些额外的信息。
- 每个区块B^r对应一个余额表，用S^r表示，S^r记录了系统在区块B^r时的所有账号信息。S^0由B^0可以计算得到，S^r可以由B^r和S^r-1共同计算确定。
- 余额表S^r包含了系统中每个用户的账户详情。账户详情包含了以下的信息：
  - APK_u：账号公钥（拥有该账户的用户保管与之相对应的私钥）
  - b(r,u)：用户u在第r轮拥有的余额
  - 用户的状态
- 用户的账号详情还包含了一些额外信息，这些信息可以在用户参加共识协议时使用：
  - VRFPK_u：VRF算法的公钥，主要用来做秘密抽签
  - PartPK_u：参与签名的公钥
  - [v,w]：在第v轮到w轮，用户有权参与共识。
- 关于用户状态：
  - 一个用户在第r轮处于在线状态需要满足以下几个条件：
    - 在第r-322轮对应的余额表中该用户的账户详情中，VRFPK_u是非空的
    - 该用户的PartPK_u在第r轮是合法的，也就是说r>=v，r<=w，PartPK_u非空
- 最小账户余额

    - 用户的账户中最小余额是0.1 Algo（Algo是系统的货币符号）
- Algo精度
    - 1个micro Algo相当于10^-6次方Algo（类似人民币中1分等于10^-2元）

2、Key管理

- 多重签名地址
    - Algorand是支持多重签名的。例如用户创建一个由3个私钥生成的多重签名地址，设置threshold为2，就表示只要3个私钥中有任意2个私钥的签名通过就可以了。为了满足这一需求，需要：
        - 生成3个key，分别是(PK1,SK1)，(PK2,SK2)，(PK3,SK3)。
        - 计算账户地址，APK = Hash(PK1,PK2,PK3,threshold=2)。
    - 现在，假定用户u的多重签名地址APK对应的账号有一定数量余额。APK对应的一笔转账交易中的签名至少需要包含2个或以上的私钥签名信息，即SK1，SK2，SK3中任意两个对这笔交易的签名信息。
- 可验证随机函数（VRF） Keys
    - 用户u需要在自己的账户信息表中添加一个VRF公钥（VRFPK_u）来参与共识协议，与这个公钥相对应的私钥（VRFSK）由用户秘密保管。我们使用符号VRF(x)来表示私钥VRFSK对信息x的签名。
    - Algorand使用的VRF算法是可以防止恶意注册key的。即使一个用户在链中恶意生成注册了一个VRFPK，所有的输出依然是独一无二，无偏差的。原因是：
        - VRF使用的是第r-2轮的种子作为输入
        - VRFPK必须在第r-322轮已经在系统中注册过
        - APK是用户的唯一公钥
- 参与方的Keys
    - Algorand使用Ed25519来生成多个参与的key对
    - 如果用户想参与共识协议：
        - 先生成一个公私钥key对，表示为(PartPK_u,PartSK_u)
        - 生成在[v,w]之间的一组临时公私钥key对（总共w-v个），表示为(EPK^v,ESK^v,.....,EPK^w,ESK^w)
        - 使用PartSK_u对EPK^i和轮数i进行签名得到sigma_i，直到签名完所有生成的EPK，删除保存的PartSK_u。生成的集合表示为((EPK^v,ESK^v,sigma_v),.....,(EPK^w,ESK^w,sigma_w))
        - 使用ESIG^r(x)表示ESK^r对信息x的一个签名。参与共识之前，用户需要注册PartPK_u，[v,w]以及每一轮对应的临时key到系统中。
        - 假设用户可以参与本轮共识，在第r轮签名消息M的流程如下：
            - 使用私钥ESK^r签名消息M得到sigma_M
            - 广播消息(sigma_M,EPK^r,sigma_r)到网络中。

        - EPK^r可以用于验证sigma_M，通过验证sigma_r确定PartPK_u对EPK^r的所有权
        - 区块B^r被确认之后，用户删除本地保存的ESK^r

3、秘密选举

- 对每一轮r，Algorand执行一个快速的，partition-resilient的拜占庭共识协议。如果区块拥有一个与之相关联的证书，那么它就被最终确认了。一个区块的证书包含了一组来自委员会成员（随机秘密选举出来的）标识自身身份的合法证明以及他们对这个区块的签名。接下来，我们会说明一下随机数如何生成以及秘密选举如何进行。
- 随机种子Q^r。
    - 每个区块B^r都包含一个种子Q^r，这个种子可以用于未来某一轮共识协议的委员会选举中。Q^r+2（也就是第r+2轮的种子）是由B^r+2这个区块的矿工根据第r轮的种子Q^r和矿工的VRF keys计算生成。
    - 对于创世区块，我们设置Q^0=H(S^0)，这里函数H(?)是系统中的一个哈希方法。未来每一轮的种子都会被不同的矿工重新计算随机，从而保证了活跃度和安全性。
- 通过VRF算法来秘密选举委员会
    - 共识协议的每一轮由多个阶段组成（每个阶段都尝试达成共识），每个阶段又会被分成多步。
    - 在每一步，系统的每个人结合第r-2轮的种子Q^r-2和第r-322轮的账号信息表，通过特定算法就可以计算出自己是否属于当前这一步对应的委员会。在每一轮r，相加第r-322轮的所有在线账户的余额得到这一轮的Algo总额，我们可以称之为StakeOn_r-322。
    - 为了确定在第r轮的第p个阶段的第s步是否入选了这一步的委员会，首先，在线用户需要使用VRF函数（需要这几个参数，VRF私钥，第r-2轮的种子Q^r-2，轮数r，阶段号p，步数s），生成一个结果我们称之为output。接着，把output，用户在r-322轮的余额，StakeOn_r-322这三个参数传入另一个特殊函数，计算出output是否足够小，足够小到该用户可以成为这一步的委员会成员。
    - 委员会选举是公平的。对每一个在线用户而言，其在第r-322轮拥有的资产会对应相应数量的彩票，而每张彩票对应一个固定的能赢（入选委员会）的概率。拥有100个货币的用户相当于拥有100张彩票，这些货币可以被一个账户完全持有，也可以10个甚至100个账户分散持有，但不管集中还是分散，这个用户都无法在委员会选举中获取任何优势。
    - 在第r轮使用第r-322轮的账户信息表的部分原因是，要保证用户在看到种子之前已经将自己的VRF Keys注册到系统中了。（否则用户可能会在提前看到种子之后，选择对他选举有利的VRF key）
- 计时器和时钟
    - 每个用户u都会在本地保存一个计时器，称之为timer_u。Algorand会假设所有用户的计时器并不是同步的。timer_u也可能被用户自己重置，查询计时器时，它会返回自上一次重置开始已经经过的时间。计时器只应用于运行共识协议（例如，一个用户需

要决定进入共识协议的下一步的时机）。共识协议假定用户的计时器是以相同的频率在运行的。

4、安全模型

- Algorand中选取的参数是基于几大假设，分别是，节点大部分是诚实的，足够安全的加密算法，消息时延存在上限
- 主要诚实节点假设
  - 对每一轮r，第r-322轮中的用来选举的质押资产总和的百分之八十是诚实的。（如果我们使用更小的诚实节点占比，就需要使用更大的委员会成员数量）
- 加密充分（不被破解）假设
  - Algorand使用数字签名来做消息认证。在共识协议中使用足够随机的VRF和哈希算法。
  - 如果黑客要破解系统，需要运行高达2的128次方的哈希运算，但是这种运算所需要的时间会超过整个系统的生命周期。如此巨大的运算量，黑客可能会发现SHA256哈希函数的有限次碰撞或者在今天的算力下挖1000亿年的比特币区块。
- 避免被逐步腐蚀
  - 在Algorand协议中，用户每一轮都会改变自己参与的key。例如，在第r轮用户签名完消息之后，他就会删除这个临时key，并在未来的轮次中使用最新的临时key。这样就保证了安全性避免被黑客逐步腐蚀，如果key是固定不变的，黑客就可以在看完一个用户向网络中广播的消息之后实施攻击腐蚀。（前面说过，用户使用自己的VRF keys执行了秘密选举，在第r轮之前黑客都不知道该去腐蚀哪个人）
  - 还有，即便发生最坏的情况，黑客腐蚀掉了委员会的所有成员。只要拥有大部分投票权的用户是诚实的，可以删除他们的临时keys，在同一轮不会有两个独立的合法区块出现。

5、交易

- 转账交易
  - 用户可以执行从一个账户到另一个账户的资产转账交易。任何一笔交易都需要发送者的私钥签名才有效。
- 账户状态转变：在线或不在线
  - 用户可以发送一类交易在区块链中注册一个VRF公钥和参与签名的key，用于参与共识协议。如果用户在某一轮注册过用于签名的合法key，那就认为用户在这一轮是在线的。
  - 还有一种类似的交易，用户可以改变自己的状态为离线。离线用户无权参与共识。
- 任何一笔在网络中传输的交易需要指定合法的轮数区间和交易手续费
  - 交易合法轮数区间
    - 任何一笔交易都需要包含一个合法的轮数区间[r1,r2]。交易只有存在于此区间内的区块中才是合法的。这个区间被用来判断交易是否已经出现在区块链中：用户只需要回顾[r1,r-1]之间的区块就可以判断交易在区块r中是否合法。

- 交易手续费
  - 每笔交易必须包含至少0.001 Algos的手续费。交易手续费的存在可以防止黑客发送大量的垃圾交易。

6、激励

- 在每一轮r，根据用户在第r-322轮的余额占比，系统会按比例派发给用户奖励。不论用户在第r轮或者第r-322轮是否在线，都会获得奖励。这笔奖励由一个被叫做RewardsPool的账户分发，这个账户不能用于任何其他目的。任何账户都可以向RewardsPool转账，同时Algorand基金会也可以自行决定向此账户转账。每过50万轮，RewardsPool账户的余额被用来计算未来50万轮每一轮派发出去的奖励金额。例如，假定当前余额是b，那下一个50万轮，每一轮派发的奖励就是(b-0.1)/500000。其中0.1是账户的最低余额。
- 在系统中，用户收到的奖励会不断累积并且是实时可用的。当用户u参与了任何一种类型的交易，在第r轮，奖励会加到他上次发生变动的余额中。在第r轮之后，这个新余额会参与分配r+322轮之后的奖励。
- 如果用户在第r轮关闭了自己的账户，那么他将不会受到r,r+1,.....,r+321轮的奖励。

7、共识协议

每一轮r都由一个或多个阶段组成。在任意一个时间点，一个用户u必定是处于某一轮的某个阶段，可以记为(r,p)。一个阶段(r,p)又可以分成多步，记为(r,p,s)，s的范围是0到255。用户通过本地的计时器timer_u，来决定何时进入当前所处阶段的哪个步骤step。当用户进入一个新的阶段时，计时器就会被重置。

7.1 随机选举委员会

- 每一步(r,p,s)都会产生一个委员会来参与投票。每一步(r,p,s)都有两个仅与此步骤相关联的参数：
  - CS_s：该步骤产生的委员会成员的数量
  - CT_s：该步骤委员会数量阈值。投票超过此阈值即达到法定人数，表示通过决议。
- 对每一步，下表给出了步骤的名字和相关联的参数值

Step number s	Step Name	CS_t	CT_s
0	propose	20	N/A
1	soft	2990	2267
2	cert	1550	2267
$3 \leq s \leq 252$	next_{s-3}	5000	3838
253	late	500	320
254	redo	2400	1768
255	down	6000	4560

7.2 描述投票过程的符号

- StakeOn_r-322：在第r-322轮的所有投票权总和（可以认为是所有用户资产的总和）

- q

$$q_s^r = \frac{CS_s}{StakeOn_{r-322}}$$

- (r,p,s)-credential：VRF(Q^r-2,r,p,s)，如果n大于等于1，表示这个用户有权参与投票
- (r,p,s)-credential weight：如果n使得下面的不等式成立并且n大于等于1，表示用户有权参与投票：

$$\sum_{k=n+1}^{\infty} Binomial(b_u^{r-322}; q_s^r) < VRF_u(Q^{r-2}, r, p, s) \leq \sum_{k=n}^{\infty} Binomial(b_u^{r-322}; q_s^r)$$

- (r,p,s)-credential value：用n表示凭证中的权重，在0到n的区间可以找到一个值记为i（其中h_i是(credential,i)的哈希值），使得h_i最小。
- (r,p,s)-credential vote：
- (r,p,s)-credential quorum：

7.3 时间参数

- λ：在网络状况良好的情况下广播一段小数据量网络包（例如一个投票）所需的时间
- Λ：在网络状况良好的情况下广播一段大数据量网络包（例如一个区块）所需的时间
- λ_f：

7.4 协议的详细说明

- 当用户收到了一条消息，消息表明在第r-1轮对一个区块B^r-1的投票已经满足了cert这一步的法定人数，那么就可以开始第r轮阶段0了。当用户处于第r轮的第p个阶段，收到一条消息，表明投票在第p`阶段的next这一步达到了规定的法定人数（其中p`>=p），用户就可以进入到第p`+1阶段了。
- 每当用户进入一个新的阶段甚至新的步骤，都需要重置计时器timer_u，这样可以明确在每一步投票的时机。
- 阶段(r,p)的投票过程说明：
  - 当用户u进入阶段(r,p)时，他已经完成了p之前的所有阶段并且重置计时器从0开始。
  - 在每一步，任何一条广播的消息都包含了用户的签名和用户针对该步生成的凭证。
  - 接下来，只是简单描述每一个步骤，而忽略了具体的实现过程：
    - STEP 0: [Proposal阶段] 当计时器为0时：
      - 如果p是0，或者在p>0时并且收到在p-1阶段的next步对空块的投票达到法定人数的消息，那么用户打包一个新的区块并广播区块和区块的哈希

值到网络中
    - 否则，如果p>0并且收到在p-1阶段的next步对某非空块的投票达到法定人数的消息，则用户广播此非空区块
- STEP 1: [Filtering阶段] 当计时器为2λ时：
    - 如果p是0，或者p>0时并且收到在p-1阶段的next步对空块的投票达到法定人数的消息，那么用户从自己本地的区块列表（包括从网络中收到的和自己生成的）中选择对应凭证最小的区块，并且对该区块投票
    - 否则，如果p大于0并且收到在p-1阶段的next步对某非空块的投票达到法定人数的消息，那么用户对这一区块投票并广播
- STEP 2: [Certifying阶段] 当计时器大于2λ小于max(4λ,Λ)时：
    - 如果用户对某个区块的软投票（也就是step1对区块的投票）达到了法定人数，那么用户对该区块进行认证投票（对投票再投票）
- STEP 3到252: [Recovery阶段] 当计时器=max(4λ,Λ)，此时s=3，或者，计时器=max(4λ,Λ)+(2^s-3)λ+r，此时s大于等于4小于等于252，同时r从0到(2^s-3)λ之间随机取值
    - 如果用户收到一个区块并且收到在p阶段的soft步对这个区块的软投票（STEP1的投票）达到法定人数的消息，则进行next投票
    - 否则，如果p>0并且收到在p-1阶段的next步对空块的next投票（STEP3到252的投票）达到法定人数的消息，则对空块进行next投票
    - 否则，如果用户收到一个区块并且收到在p-1阶段的next步对某非空区块的next投票（STEP3到252的投票）达到法定人数的消息，则对该区块进行next投票
- STEP 253到255:[Fast recovery阶段] 当计时器=kλ_f+t，其中k是任意的正整数，t从大于等于0小于等于λ_f的区间随机取值
    - 如果用户收到一个区块并且收到在p阶段的soft步对这个区块的软投票（STEP1的投票）达到法定人数的消息，则进行late投票
    - 否则，如果p>0并且收到在p-1阶段的next步对空块的next投票（STEP3到252的投票）达到法定人数的消息，则对空块进行down投票
    - 否则，如果用户收到一个区块并且收到在p-1阶段的next步对某非空区块的next投票（STEP3到252的投票）达到法定人数的消息，则对该区块进行redo投票

7.5 种子

- 在区块B^0，Q^0=H(S^0)。
- 前面提到过S^-r=S^0，Q^-r=Q^0。在第r轮第p个阶段的合法区块中的种子Q^r符合如下规则：

$$
\begin{aligned}
&\text{If } r \equiv 0 \bmod 160 \text{ or } r \equiv 1 \bmod 160\text{:}\\
&\quad \text{If } p = 0\text{:}\\
&\quad\quad Q^r = VRF_u(Q^{r-2}, H(B^{r-160}))\\
&\quad \text{Else:}\\
&\quad\quad Q^r = H(Q^{r-2}, H(B^{r-160}))\\
&\text{Else:}\\
&\quad \text{If } p = 0\text{:}\\
&\quad\quad Q^r = VRF_u(Q^{r-2})\\
&\quad \text{Else:}\\
&\quad\quad Q^r = H(Q^{r-2})
\end{aligned}
$$

- 7.6 安全特性

  下面我们会列出协议为了保证安全的一系列特性。每个特性都对应每个阶段的法定人数限制和步骤。我们列出了每个阶段相关联的步骤。第4和第5中条件是相似的，分开他们是为了区分相对应的不同步骤。

  - 1、恶意用户无法在任意一步组成一个合法的委员会。相关联步骤：1到255步
  - 2、不会在soft步骤出现两个不同的合法委员会。相关联步骤：第1步
  - 3、如果在cert阶段对某个区块达成共识，那么就不会在down阶段和next阶段对一个空区块达成共识。相关联步骤：第2步到第252步，第255步
  - 4、如果在soft步骤对某个区块达成共识，那么就不会在next阶段对另一个不同的区块达成共识。相关联步骤：第1步，第3步到第252步
  - 5、如果在soft步骤对某个区块达成共识，那么就不会在redo阶段对另一个不同的区块达成共识。相关联步骤：第1步，第254步

- 8、协议更新

  Algorand基金会管理区块链的开发。通过以下几个步骤完成区块链的核心模块更新：

  - 1、Algorand基金会会在一个公开的git仓库公布协议的规范说明。仓库提交的URL被当做”协议版本“的一个特殊标识。这个URL必须包含关联git提交的hash值。
  - 2、任意协议相关的更新必须通过社区的矿工们投票支持。一个矿工在打包的区块中加入该更新的版本号表明支持此次更新。如果在10000轮中有8000轮已经被确认的区块中包含了此次更新的版本号，就可以认为该更新被社区认可了。
  - 3、在经过了10000轮投票之后，普通用户也可以更新自己的软件版本了。最新的规范也就生效了，从此基于最新的规范生成区块。

- 参数表

Committee 0 size (CS_0)	20
Committee 1 size/threshold (CS_1/CT_1)	2990/2267
Committee 2 size/threshold (CS_2/CT_2)	1500/1112
Committee $3 \leq s \leq 252$ size/threshold (CS_s/CT_s)	5000/3838
Committee 253 size/threshold (CS_{253}/CT_{253})	500/320
Committee 254 size/threshold (CS_{254}/CT_{254})	2400/1768
Committee 255 size/threshold (CS_{255}/CT_{255})	6000/4560
Minimum account balance	0.1
Minimum transaction fee	0.001
Smallest Algo amount	10^{-6}
Seed lookback	2
Parameter refresh interval	160
Seed lookback	320
$\lambda(seconds)$	2
$\Lambda(seconds)$	17
$\lambda_f(seconds)$	300