Sperax: An Approach To Defeat Long Range Attacks In Blockchains

Yongge Wang
UNC Charlotte
Charlotte, NC, USA
yonwang@uncc.edu

Jingwen Sun, Xin Wang, Yunchuan Wei, Hao Wu
Sperax
Beijing, China

Zhou Yu
Mathematics
UC Los Angeles
yuzhou87@g.ucla.edu

Gillian Chu
EECS
UC Berkeley
gillichu@berkeley.edu

***Abstract*—Since Bitcoin's seminal work of achieving consensus under the assumption that more than 51% computational power is honest, many researchers have proposed various kinds of consensus protocols. In recent years, there has been a trend in designing proof of stake (PoS) based consensus protocols. Unfortunately, PoS based consensus protocols are inherently not secure and are vulnerable to *"long range attacks"*. Thus PoS based blockchains have the potential risk of being forked with a minimal cost for double spending. In this paper, we propose a PoS protocol that is secure against long range attack based double spending. The security is achieved using secure randomness beacons generated by tamper proof hardware modules.**

***Index Terms*—PoS blockchain, Byzantine Fault Tolerance, authenticated randomness**

I. Introduction

In a digital society, it would be convenient to have a digital transaction system or to have a digital currency system. It is generally easy to design an electronic cash system using public key infrastructure (PKI) systems. But PKI-based electronic cash is also easy to trace. Theoretically, banknotes could be traced using sequence numbers, though there is no convenient infrastructure to trace banknote sequence numbers back to users. Thus banknotes maintain sufficient anonymity.

Several researchers have designed anonymous electronic cash systems. (see, e.g., Chaum [2]). However, these systems have not attracted enough interest from the society and they have never been adopted. The situation has changed since the cryptographic currency Bitcoin was introduced in the paper [4] by a pseudonym "Satoshi Nakamoto". Since 2009, the implementation of Bitcoin has been in operation and it has been widely adopted as one of the major cryptographic currency on the market now. Bitcoin used Forth-like Scripts for writing smart contracts. In order to increase the smart contract capability, Ethereum used Turing-complete programming language *Solidity* for its smart contract design.

In the Bitcoin system, one can achieve system consensus under the assumption that more than 51% computational power is honest. This "contradicts" the classical results in Byzantine Agreement which requires at least 2/3 of the participants to be honest for achieving consensus. Specifically, Garay et al [3] proved that if the adversary's hashing power is at most $\frac{t}{n}$, then the adversary controls at most $\frac{t}{n-t}$ percentage of the blocks in the blockchain (the authors also showed that a self-mining strategy could be used to achieve this bound). For example, if $t < \frac{n}{3}$, then the adversary controls at most 50% of the blocks in the honest players' blockchain. However, Bitcoin has several inherent technical challenges. First, Bitcoin uses proof of work to generate new blocks. This requires a lot of computation and wastes a lot of energy. Secondly, due to the enormous amount of computational power and energy requirements, it is not profitable for regular desktop computers to mine new Bitcoin blocks. Thus the major hasing powers for Bitcoin block generation are currently from a few mining pools (in particular, Chinese mining pools control more than 75% of the Bitcoin network's collective hashrate in 2019) and the assumption of honest majority computing power may no longer be valid. Thirdly, Bitcoin block chain may fork once a while. Thus one needs to wait for a few blocks to make sure that her transaction becomes permanent on the block chain.

The ingenious design of Bitcoin has inspired a lot of fruitful research. Several researchers have introduced proof-of-stake or proof-of-"something" block chain techniques to address the challenges faced by Bitcoin. However, as noted in Poelstra [5], it is impossible to design secure distributed consensus without consuming some resource outside of the system. This is generally called the *"costless simulation attack"* or *"nothing at stake attack"*. A special kind of costless simulation attack is the *long range attack*. If it is costless for a block proposer to create valid blocks, then she may sell her stake in the system and perform a history-rewrite attack. For example, for a set C of nodes that controls the majority of the stake at block r, they can collude to fork the history of the blockchain from the block r. In this alternate history, transaction could be carried out in a way that C continues to hold majority stake. The literature recommends the use of "bootstrapping through social consensus" approach to defeat "costless simulation attack". In this bootstrapping approach, it is recommended that, for a new node to join the blockchain system, one should manually provide a recent checkpoint blockchain from the set of nodes it can access upon joining. In this paper, we propose a new consensus protocol assuming the existence of tamper proof devices that generate authenticated pseudo-random sequences. Our protocol can be considered a proof-of-stake protocol where the stake is the posession of a hardware module. Since the tamper proof hardware modules could be designed in such a way that they contain counters for signed blocks, the tamper proof hardware module based proof-of-stake protocols can defeat double-spending based on costless simulation attacks. Up to our knowledge, our protocol is the first proof-of-stake protocol that is secure against double-spending based on costless simulation attacks (without manual bootsrapping). On the other hand, our protocol does not prevent general costless simulation attacks from participants who have never made any transactions. But there is no motiviation for these users to launch the costless simulation attacks.

II. Tamper-proof hardware modules

Random sequences are important for achieving blockchain security. In the SperaX PoR blockchain, the randomness is generated using tamper proof hardware modules. There has been an extensive research on using tamper-proof hardware modules to achieve a variety of cryptographic goals. In this paper, we use these hardware modules to establish random beacons for proof-of-stake blockchain design. Our system is called a proof-of-authenticated-randomness (PoR) based consensus protocol where each participant has access to a SperaX certified random number generator. To prevent adversaries from retrieving keys or modifying operations in the random number generator, the random number generator chips are designed with tamper-proof properties. A SperaX tamper-proof chip is designed to zeroise its sensitive data if it detects penetration of its security encapsulation or out-of-specification environmental parameters. Tamper proof hardwares range from cheap smart cards to expensive CPUs such as Intel SGX. SperaX random number generators should use at least FIPS 140-2 Level 3 approved hardwares.

There are various techniques for generating random numbers from physical processes such as using microscopic phenomena (e.g., statistically random thermal noise, the photoelectric effect, involving a beam splitter, and other quantum phenomena). However, not all of these microscopic phenomena may be made tamper-proof. For example, if the tamper proof chips use thermal noise, it is important to make sure that an adversary may not be able to disturb such kind of processes by changing the environments of the running chips. In addition to microscopic phenomena based random generators, one may also use cryptographic pseudo-random generators for PoR consensus protocols. For example, in an interactive pseudo-random generator, when user input a binary sequence input e, the pseudo-random generator may output $H(key, e)$ where H is random oracle (e.g., a secure cryptographic hash function) and key is a secret stored in the tamper proof hardware module. For SperaX blockchain system, it is recommended to use NIST SP800-90A Deterministic Random Bit Generators (DRBG) to implement the random oracle in the tamper-proof hardware modules.

There is a certificate authority for SperaX hardware. Each tamper proof hardware module contains a random number generator and the its private key (together with SperaX certified public key). The hardware module that we will use is an interactive one and has two kinds of functionalitis.

1) Authenticated randomness generation: When a user P_i inputs a blockchain round number r and step number s to her hardware module, the hardware module outputs a digitally signed random number $R_{i,r,s}$ together with the singature $(r, s, R_{i,r,s}, SIG_{cpk_i}(P_i, r, s, R_{i,r,s}))$ where cpk_i is the SperaX certified public key for user P_i's hardware module.
2) Transaction confirmation: the hardware module holds a counter number C_i^B which means that all transcactions for the user P_i on the blockchain before block C_i^B have been certified by the hardware module. Initially the counter $C_i^B = 0$. If the user P_i has carried out a sequence of transactions in blocks $B^{r_1}, .., B^{r_a}$ where $r_1 < r_2 < \cdots < r_a$ and the transactions in B^{r_1} have not been certified by the hardware module yet, the user inputs $r_1, \cdots, r_a$ to the hardware module. The hardware module outputs the digital signature $(H(r_1, \cdots, r_a), SIG_{cpk_i}(P_i, H(r_1, \cdots, r_a)))$ where cpk_i is the SperaX certified public key for user P_i's hardware module. At the same time, the hardware module sets the new counter value as $C_i^B = r_a - 1$. That is, all transactions in $B^{r_1}, .., B^{r_a - 1}$ have been certified by the hardware module and it will not certify any blocks before $C_i^B = r_a - 1$ any more.

III. PoR Consensus Protocol

In our design, each participating node needs to have SperaX certified tamper-proof hardware module. By this requirement, PoR can be considered as a permissioned blockchain. However, we assume that anyone can freely buy a SperaX certified tamper-proof hardware module on the market anonymously, PoR can also be considered as a permissionless blockchain. The initial status of the block chain is

$$S^0 = \{(P_1, a_1), \cdots, (P_j, a_j)\}$$

where $P_1, P_2, \cdots, P_j$ are a list of initial users and $a_1, \cdots, a_j$ are their respective initial amounts of money units. We assume that each user P_i is identified by its public key pk_i. That is, for the users $P_1, P_2, \cdots, P_j$, their corresponding public keys are $pk_1, \cdots, pk_j$. In practical implementations, a user P_i may be identified by the hash of her public key. That is, we may use $P_i = H(pk_i)$ in implementations. When a new user joins the network, she needs to register her hardware module using her public key pk_i in the blockchain. A user should have a unique hardware module registered in the system. In other words, a user may be uniquely identified by her public key or by her hardware module. A valid transaction from a user P_i to a user $P_{i'}$ is in the format of

$$SIG_{pk_i}(P_i, P_{i'}, a')$$

where the user P_i currently has $a \geq a'$ money units, $P_{i'}$ is an existing or a newly created user, and pk_i is the public key of user P_i. The impact of this transaction is that the amount of money units for user P_i is decreased by a' and the amount of money units for user $P_{i'}$ is increased by a'.

In an idealized magic ledger system, all transactions are valid and the list L of sets of transactions are posted in a tamper-proof box in the sky which is visible to all participants

$$L = TX^0, TX^1, TX^2, \cdots,$$

PoR block chain is organized in a series of rounds $r = 0, 1, 2, 3, \cdots$. Similar to the initial status, the system status for round $r > 0$ is a list of users and their corresponding money units

$$S^r = \left\{(P_1, a_1^{(r)}), (P_2, a_2^{(r)}), (P_3, a_3^{(r)}), \cdots\right\}$$

In a round r, the system status transitions from S^r to S^{r+1} via the transaction set TX^r

$$TX^r : S^r \to S^{r+1}.$$

In PoR, the blockchain is a list of blocks $B^0, B^1, \cdots, B^r$ where each B^r consists of the following fields: the block number r itself, the time-stamp t_r that the block B_r is generated, the set TX^r of transactions for round r, the hash of the previous block $H(B^{r-1})$, the user P^r who generates this block, and a set $CERT^r$ of signatures certifying that the block B^r is constructed appropriately

$$B^r = \left\{r, t_r, TX^r, H(B^{r-1}), P^r, CERT^r\right\}.$$

The field $CERT^r$ is a list of signatures for the value $H\left(r, t_r, TX^r, H(B^{r-1}), P^r\right)$ from more than 2/3 of the members of the verifier committee V^r for round r.

In PoR, we assume that all messages are timely delivered in the entire network. Specifically, we assume that, at the start of round r, all users should have learned the current blockchain $B^0, B^1, \cdots, B^{r-1}$. From this chain, one can deduce the user sets $U^0, U^1, \cdots, U^{r-1}$ and their corresponding money units of each round. In order for a user to serve as a potential leader or as a verifier committee member, she must have joined the system at least κ-block before where κ is a system parameter. Normally we can take $\kappa = 3$. The choice of $\kappa > 0$ will guarantee that the system is running in a consistant way.

For a user P_i with a_i amounts of money units, she should be considered as a_i users $(P_i, 1), \cdots, (P_i, a_i)$ during the leader and verifier committee selection process. Let Ω_r be the total amount of money units at the round r where the leader or the verifier committee is selected. Specifically, a potential leader of round r is a user P_i if $P_i \in U^{\max\{0, r-\kappa\}}$, and

$$0.H(r, 1, R_{i,r,1}, W_r) > (1-p)^{a_i} \tag{1}$$

where $R_{i,r,1}$ is the random sequence output of user P_i's hardware module with input $(r, 1)$ (that is, round r and step 1), and W_r is defined as follows:

1) $W_0 = H(U^0)$,
2) $W_j = H(P^{j-1}, W_{j-1})$ for $0 < j \le r$ and $P^0 = \emptyset$,

The value p in the formula (1) is a pre-determined probability chosen in such a way that, with overwhelming probability, at least one potential leader is honest and online. For example, one may choose $p = \frac{20}{T_r}$ where T_r is the expected total number of money of active online users in the system during the round r. Since the user P_i cannot change the output of her tamper proof hardware module, she cannot modify the value $R_{i,r,1}$. Thus it is guaranteed that a user P_i cannot increase her probability to be a leader by changing the value $R_{i,r,1}$. We also note that the user P_i is the only person in the system that can determine whether she is a potential leader since she is the only person that holds the value $R_{i,r,1}$ from her hardware module. Other users can verify that the user P_i is a potential leader after P_i discloses the credential $SIG_{cpk_i}(P_i, r, 1, R_{i,r,1})$ on $R_{i,r,1}$. The leader P_{l^r} is defined to be th user whose hashed random string is the largest. That is,

$$0.H(r, 1, R_{l_r,r,1}, W_r) \ge 0.H(r, 1, R_{i,r,1}, W_r) \tag{2}$$

for all potential leaders P_i.

At the start of round r, each potential leader P_i of round r collects the maximal transaction set TX_i^r of round r that have been propagated to her. Then she computes the candidate block B_i^r without the certificate $CERT^r$

$$B_i^r = \left\{r, t_r, TX_i^r, H(B^{r-1}), P_i\right\}.$$

The user P_i then propagates the message $m_i^{r,1} = (B_i^r, SIG_{cpk_i}(P_i, r, 1, R_{i,r,1}))$ to the entire network.

Since there could be several potential leaders during round r, each user could receive several candidate block messages $m_i^{r,1}$ from the preceding paragraph. Thus we need to select a verifier committee to determine the actual leader P_{l^r} and finalize B^r as the corresponding block $B_{l_r}^r$ proposed by this leader. The verifier committee is selected as follows. A user P_i belongs to the verifier committee V^r for $\tau > 0$ times (that is, P_i's signature will be counted as τ signarures) if $P_i \in U^{\max\{0, r-\kappa\}}$, and

$$\begin{array}{l}\sum_{j=0}^{\tau-1} \binom{a_i}{j} (p')^j (1-p')^{a_i - j} \le 0.H(r, 2, R_{i,r,2}, W_r) \\ \quad < \sum_{j=0}^{\tau} \binom{a_i}{j} (p')^j (1-p')^{a_i - j}\end{array} \tag{3}$$

where $R_{i,r,2}$ is the random sequence output of user P_i's hardware module with input round r and step 2, W_r is recursively defined in the same way as in the leader selection process, and p' is a pre-determined probability. For example, one may choose $p' = \frac{1000}{T_r}$ where T_r is the expected total number of money of active online users in the system during the round r. It should be noted that in the formula (3), the value $\binom{a_i}{j} (p')^j (1-p')^{\omega_i - j}$ is the probability that the user P_i is selected for exact j-times if each individual sub-user (P_i, j') is selected with the probability $\frac{p'}{\Omega_r}$.

Each verifier P_i in V^r determines that the user P_l is the round leader if the conditions in both the formula (1) and the formula (2) are satisfied for all potential leader P_i contained in the messages $m_i^{r,1}$ that she has received.

After verifying the validity of the message $m_l^{r,1} = (B_l^r, SIG_{cpk_l}(P_l, r, 1, R_{l,r,1}))$, the verifier P_i authenticates and propagates her candidate block B_l^r to the entire network using the signature $SIG_{pk_i}(\tau_i, H(B_l^r))$ where τ_i is the number of time that the user P_i should serve in the verifier committee.

Next each user P_i checks whether she has received more than $\frac{2v_r}{3}$ signatures for some candidate block B^r where v_r is the total number of committee members of V^r (note that a committee member may be counted multiple times). If there exist more than $\frac{2v_r}{3}$ signatures for a proposed block B^r, then P_i marks B^r as the final round r block. If no more than $\frac{2v_r}{3}$ signatures for a proposed block is received, then P_i marks the block B^r as an empty block. It is straightforward to see that if the majority moneys are honest, then all the honest nodes will reach an agreement on the block B^r at the end of round r.

Generally a more involved Byzantine Agreement protocol could be used to offer rewards for those verifier committee members who have served. However, it may not be worthwhile to add this extra layer of operations to motivate verifier committee members. One may assume these committee members would liketo work voluntarily to keep the system in function.

IV. Defeating double spending attacks

In the double spending attacks, an adversary $\mathcal{A}$ may corrupt all users in a block B^r and then fork the chain from the block B^r and on. The adversary's chain could be generated more efficiently and overpass the honest chain since the adversary will only include simple transactions in each block. It is not hard to see that these attacks will also work against our initial consensus protocol PoR introduced in Section III. In this section, we show that if we require that each transaction by a user P_i be certified by the user P_i's hardware module, then the protocol PoR is secure against costless simulation based double spending attacks. That is, a participant will not be able to fork the blockchain before a block B^r if she has carried out transactions in or after block B^r. However, our protocol will not be able to prevent the participants to fork the chain if they have never carried transactions on the chain. It is our belief that there is no motivation for these participants to fork the chain though.

As we have mentioned in the proceding sections, each hardware module maintains an internal tamper-proof counter C_i^B. The hardware module would not certify any transactions contained in blocks B^r if $r < C_i^B$. Specifically, we require that each transaction in the PoR blockchain is considered *final* only after the payer's hardware module has certified the transaction.

Let $B^{r_1}, \cdots, B^{r_a}$ be a sequence of blocks where the user P_i has been involved in transacations contained in these blocks, and the transactions in B_{r_1} have not been certified by P_i's hardware module. Furthermore, assume that P_i has not been involved in any transactions in blocks B^r where $r_1 < r < r_a$ and $r \neq r_j$ $(1 \leq j \leq a)$. User P_i can certify the transations in the blocks $B^{r_1}, \cdots, B^{r_a-1}$ using the following procedures. The user give the input $r_1, \cdots, r_a$ to her hardware module and the hardware module outputs the following signature

$$(H(r_1, \cdots, r_a), SIG_{cpk_i}(P_i, H(r_1, \cdots, r_a))) \qquad (4)$$

where cpk_i is the SperaX certified public key for user P_i's hardware module. At the same time, the hardware module sets the new counter value as $C_i^B = r_a - 1$. That is, all transactions in $B^{r_1}, .., B^{r_a-1}$ have been certified by the hardware module and it will refuse to certify any blocks before $C_i^B = r_a - 1$ in future. The user P_i will then submit the signature (6) to the blockchain. Only after this signatue is included in a block on the blockchain, the transactions contained in $B^{r_1}, \cdots, B^{r_a}$ are considered as final.

For the above transaction certification process, no users can make a transaction when the user does not have a linked hardware module. Since any hardware could fail, we need to make sure that a hardware failure will not void the user's property. For this purpose, we will have a process for a user P_i to revoke her hardware module and link her property to a newly issued hardware module. The speraX system should frequenlty broadcast the list of revoked hardware modules so that one can make sure that revoked hardware modules should not participate in any transactions in future.

In the following, we informally show that our above transaction certification process will defeat the costless simulation based double spending attacks. Assume that the adversary $\mathcal{A}$ has forked the blockchain from the block B^r and we have two chains $\mathcal{B}$ and $\mathcal{B}'$ forked from B^r. Furthermore, assume that a user P_i's last transaction activity before block B^r is in the block B^b where $b < r$, and assume that she tries to spend her money on both chains. For example, she tries to spend the money on block $B^{b'}$ of chain $\mathcal{B}$ and on block $B^{b''}$ of chain $\mathcal{B}'$ where $r < b'$ and $r < b''$. In order for user P_i to certify these transactions, she needs to query her hardware module on the following two inputs (b, b') and (b, b'') where $r < b'$, $r < b''$, $B^{b'} \in \mathcal{B}$, and $\bar{B}^{b''} \in \mathcal{B}'$. Now assume that her hardware module has certified the transactions contained in $B^{b'}$. That is, the query (b, b') has been made to the hardware module. Then the hardware module has set her tamper-proof counter to $b' - 1$. Now if the user tries to certify to transactions contained in $B^{b''}$, she has to submit the query (b, b'') to her hardware module. Since $b \leq b' - 1$, the hardware module would refuse to output a signature. Thus the PoR system is robust against costless simulation based double spending attacks.

V. Security analysis

In the following paragraphs, we carry out a formal analysis on the probability at which malicious nodes can make the first block when attempting to make a fork. The analysis will show that this probability is practically low enough when appropriate SperaX-Foundation-controlled parameters are chosen, so SperaX blockchain is safe against such kind of attacks. First of all, we want to formally introduce the methodology by which we will do the analysis.

A. Methodology

Assumptions. (i). We assume that the whole SperaX blockchain system has just completed round $r - 1$, and that all active nodes are now working on the r^{th} block, and that offline nodes neither do any work nor give any response. (ii). We assume that when attempting to make a fork at round r, malicious nodes partition the whole network by separating themselves from the rest of the nodes. That is, from their point of view, all other nodes (both active and offline nodes) are offline while from honest, active nodes' point of view, these malicous nodes are offline (although these malicious nodes are *in fact active*). Thus, a node's answer to the question that whether a particular node is active or offline depends on which partition this node belongs to. However, we assume here that the SperaX Foundation always know the *actual activity states* of all nodes. (Even if some geeky malicious nodes have succeeded in cheating on the SperaX Foundation in terms of their activeness, the Foundation always has a way to accurately estimate the total number of active nodes and total active money amount at any round.) (iii). In order to test the safety of our blockchain under the worst senario, we assume that *all* malicious nodes are active and deliberately conspire to make a fork. (iv). We assume that at any round, the malicious nodes control less than $\frac{1}{3}$ of the total money and active, honest nodes control more than half of the total money. If the network fails to meet either of these two conditions, it is considered broken and not safe anymore.

Factors. The probability $Prob_r$ for malicious nodes to successfully to fork the first block is determined by six factors:

1) Ω_r, the total amount of money at round r (including both active and offline money)
2) α_r, the ratio of the amount of money controlled by malicious nodes to the total amount of money at round r (in short, bad money over total money)
3) ρ_r, the ratio of the amount of money controlled by active nodes to the total amount of money at round r (ρ_r is also called *the participation rate at round* r. In short, it is active money over total money)
4) N_r, the numerator in the expression of p (e.g. 20 in $p = \frac{20}{T_r}$ in section 3)
5) N'_r, the numerator in the expression of p' (e.g. 1000 in $p' = \frac{1000}{T_r}$ in section 3)
6) V_{min}, the *absolute* minimum amount of votes a block needs in order to be valid which is *not* the minimum $\frac{2}{3}$ requirement

Formula. To calculate $Prob_r$, it is essential to understand two essential steps in which malicious nodes can successfully propose the first block. The first step is to have at least one malicious node elected as a potential leader so that there *is* one node that can propose a candidate block for later verification. We denote the probability for malicious nodes to succeed in this step as $Prob_r^1$. The second step in which malicious nodes have to succeed is to gather at least V_{min} votes among themselves during the verifiers election process so that they can have enough votes required to validify the candidate block proposed by one of them. We denote the probability for them to succeed in this step as $Prob_r^2$. Since these two steps are independent, the probability $Prob_r$ for malicious nodes to successfully make the first block is the multiplication of the two. That is, $Prob_r = Prob_r^1 * Prob_r^2$.

Now, in the following, we explain how P_r^1 and P_r^2 are calculated respectively. First we have

$$Prob_r^1 = 1 - (1 - p_r)^M \tag{5}$$

where M is the amount of money controlled by malicious nodes at round r (which is obtained by multiplying α_r to Ω_r and then rounding to an integer); p_r is the Foundation-chosen probability parameter as discussed in section 3 ($p_r = \frac{N_r}{T_r}$ where T_r is obtained by multiplying ρ_r to Ω_r and rounding to an integer). $(1 - p_r)^M$ is the probability of no malicious node elected as a potential leader and $1 - (1 - p_r)^M$ is the probability of having at least one malicious potential leader at round r.

We calculate $Prob_r^2$ by the formula below:

$$Prob_r^2 = 1 - \sum_{i=0}^{V_{min}-1} \binom{M}{i} (p')^i (1 - p')^{M-i} \tag{6}$$

where M is the amount of money controlled by all malicious nodes at round r; p'_r is the Foundation-chosen probability parameter as discussed in section 3 ($p'_r = \frac{N'_r}{T_r}$); V_{min} is also a Foundation-chosen parameter meaning the absolute minimum number of verification signatures needed to legitimate a candidate block. $\binom{M}{i} (p')^i (1 - p')^{M-i}$ means the probability for malicious nodes to have *exactly* i votes. $\sum_{i=0}^{V_{min}-1} \binom{M}{i} (p')^i (1 - p')^{M-i}$ means the probability for them to have *at most* $V_{min} - 1$ votes. Hence, $1 - \sum_{i=0}^{V_{min}-1} \binom{M}{i} (p')^i (1 - p')^{M-i}$ is the probability for malicious nodes to have *at least* V_{min} votes.

B. Calculation on the Worst Case

Having explained our methodology in detail above, we now calculate $Prob_r$ in *the worst case* (i) to illustrate how our methodology works and (ii) to give readers a sense of what $Prob_r$ will look like in the worst case. The list of the basic parameters needed to obtain $Prob_r$ is:

- Ω_r: 200,000,000
- α_r: 0.33
- ρ_r: **0.84 = 0.33+0.51**
- N_r: 20
- N'_r: 100
- V_{min}: 67

The following list of intermediate parameters are involved when calculating $Prob_r$:

- T_r: 168,000,000
- M: 66,000,000
- p_r: $\frac{1}{8{,}400{,}000}$
- p'_r: $\frac{1}{1{,}680{,}000}$

For the value $Prob_r^1$, we have

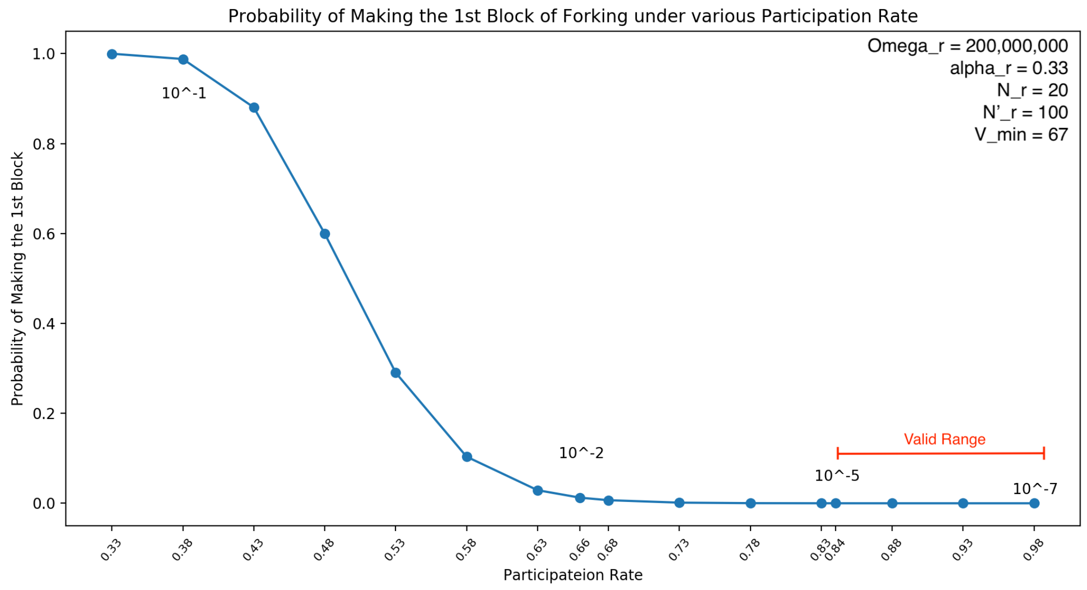
$$Prob_r^1 = 1 - \left(1 - \frac{1}{8,400,000}\right)^{66,000,000} \approx 0.9996130 \tag{7}$$

For the value $Prob_r^2$, we have

$$\begin{aligned} &Prob_r^2 = \\ &1 - \textstyle\sum_{i=0}^{66} \binom{66{,}000{,}000}{i} \left(\frac{1}{1{,}680{,}000}\right)^i \left(1 - \frac{1}{1{,}680{,}000}\right)^{66{,}000{,}000-i} \\ &\approx 3.5827e - 05 \end{aligned} \tag{8}$$

Finally, calculate $Prob_r$ as

$$\begin{aligned} Prob_r &= Prob_r^1 * Prob_r^2 \approx 0.9996130 * 3.5827e - 05 \\ &\approx \mathbf{3.5813e\text{-}05} \end{aligned} \tag{9}$$



C. Analysis

Having explained the methodology and then calculated for the worst case above, we now perform a formal analysis. The analysis will show that our blockchain is *safe* against ”network-partitioning attacks.”

We recall that $Prob_r$ is essentially determined by six factors: (1). Ω_r (2). α_r (3). ρ_r (4). N_r (5). N'_r (6). V_{min}. Firstly, among these six factors, Ω_r can be reasonably assumed to be a *constant* because Ω_r is a very large number and will not vary from rounds to rounds dramatically. Specifically, according to SperaX's economic whitepaper, we assume Ω_r to be $200,000,000$. Secondly, since it is possible that the worst senario might happen, we should make our adversary as powerful as possible when investigating the safety of our network. Thus, we assume α_r to be 0.33. This is because when more than $\frac{1}{3}$ of the money in the network are controlled by malicious nodes we simply consider our network no longer safe according to our assumptions. Thirdly, N_r, N'_r, and V_{min} are Foundation-controlled parameters. In orther words, the SperaX Foundation pre-determines some appropriate values for each of them and can adjust them at any time in order for the whole network to remain well-functioning. Specifically, as we have discussed in section 3 above, under usual circumstances, N_r is 20, N'_r is 100, and V_{min} is 67.

Therefore, since all the six factors except ρ_r are either constants or as completely controllable variables, the only source of uncertainty is ρ_r , i.e. the participation rate. Plus, given a specific set of Ω_r, α_r, N_r, N'_r, and V_{min}, if we know what $Prob_r$ will be like under every possible ρ_r, we will exhaust all the possible $Prob_r$'s in that specific setting including the worst. Since we are investigating the safety of our blockchain, the setting we are most interested in is the *worst* case under *usual* circumstance. Thus, we set Ω_r = 200,000,000, α_r = 0.33, N_r = 20, N'_r = 100, and V_{min} = 67 and draw a $Prob_r$ vs ρ_r graph as follows.

According to our assumptions, active, honest nodes control at least 51% of the total money at round r. Hence, the participation rate, ρ_r, is at least 84% (since $\alpha_r = 0.33$). As a result, in the graph above, we only need to care about the portion where ρ_r is larger than or equal to 84%. According to the graph, clearly, when ρ_r increases, $Prob_r$ goes down and our blockchain becomes safer. Hence, the largest $Prob_r$, i.e. the most unsafe, is at ρ_r = 84%. In this worst case, $Prob_r$ has its order of magnitude of *at most -5*; that is, the largest possible $Prob_r$ is less than 2^{-13} when appropriate N_r, N'_r, V_{min} are given by the Foundation (It is in fact less than 2^{-14} as our previous calculation on the worst case has shown). This number is practically low enough considering the safety of any blockchain. Therefore, based on the analysis provided above, we conclude that **our blockchain is safe against “network-partitioning attacks”** when Foundation-controlled parameters are appropriately chosen.

References

[1] M. Ahmed, J. Wei, Y.. Wang, and E. Al-Shaer. A poisoning attack against cryptocurrency mining pools. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 140–154. Springer, 2018.

[2] D. Chaum. Blind signatures for untraceable payments. In *Proc. CRYPTO*, pages 199–203. Springer, 1983.

[3] J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT*, pages 281–310. Springer, 2015.

[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[5] A. Poelstra. A treatise on altcoins, 2016. https://download.wpsoftware.net/bitcoin/alts.pdf.

[6] Yongge Wang and Qutaibah M Malluhi. The limit of blockchains: infeasibility of a smart obama-trump contract. *Communications of the ACM*, 62(5):64–69, 2019.

[7] L. Zhu, Y. Wu, K. Gai, and K. R. Choo. Controllable and trustworthy blockchain-based cloud data management. *Future Generation Computer Systems*, 91:527–535, 2019.