Sperax Blockchain: Secure BFT Consensus Protocol for Asynchronous Networks

Sperax Team

August 3, 2020

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 consensus protocols. In recent years, there has been a trend in designing proof of stake (PoS) based blockchains. Sperax is a layer-1 blockchain design that tries to address the challenges faced by many currently deployed PoS blockchains. Specifically, Sperax addresses the following challenges with its innovative designs.

1. For PoS blockchains, each block is produced by a qualified participant. In practice, there could be multiple qualified participants who would propose the next block. In order to achieve the safety property (i.e., no forks should be produced), most PoS blockchains use Byzantine Fault Tolerance (BFT) protocols to determine a unique next block. Many BFT protocols were designed with strong assumptions such as persistent secure and reliable point-to-point communication channels. These assumptions are easy to achieve in relatively closed environments though hard to achieve in open networks such as the Internet environments (e.g., at times when DoS attacks are launched). Thus these BFT protocols might not be able to guarantee the safety and liveness requirement for blockchains. Sperax leverages an innovative BFT protocol that is robust in partial asynchronous networks. It is mathematically proved that if the network eventually has synchronous periods, then the Sperax blockchain will achieve liveness and safety.
2. It is crucial to guarantee unpredictability of the participant identity who would produce the next-block. Many currently deployed PoS based blockchains do not satisfy this property though others use Verifiable Random Function (VRF) to achieve this property. Sperax adopts an innovative random beacon protocol to guarantee the unpredictability of the next-block producer identity.

In addition to these fundamental security characteristics of Sperax's layer-1 design, Sperax provides a restricted non-Turing-complete script language that could be used to develop Sperax smart contracts. Sperax smart contracts are executed on Sperax Virtual Machines (SVM). The non-Turing completeness of Sperax script language provides the feasibility of developing automatic verification mechanisms (similar to Microsoft Static Driver Verifier SDV) to verify the correctness of Sperax smart contracts. The automatic verification system provides a novel method to support various applications including finance, advertising, insurance, media, games, and other potential industries. The Sperax team has developed supporting modules that help developers to conveniently deploy their applications over Sperax blockchain. Sperax has already established initial partnership with several enterprises to deploy their business applications over Sperax blockchain.

The Sperax's innovative design provides the underlying secure infrastructure for Sperax Financial network and decentralized economy. The reader is referred to "Sperax Financial Service Network White Paper" for details regarding Sperax's global, open, instant, and low-cost payment system.

Contents

1 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. The early effort includes Chaum's online untraceable transaction system [9] based on Chaum's blind signatures and Chaum, Fiat, and Naor's [10] electronic cash that does not need the bank to be online. 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 [28] 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 [18] 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 hashing 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, there are some challenges for designing PoS blockhains as we will discuss in the next sections. Sperax blockchain tries to address these challenges.

2 Challenges of existing PoS-based blockchains

The community believes that PoS based blockchains are the future for blockchain technologies. The core technology for blockchains (including both PoW and PoS chains) is the consensus protocol. The consensus protocol determines how to generate the next block. In other words, the consensus protocol establishes a global atomic ordering for all transactions that happen in the system. If we cannot agree on the order of two transactions, then we cannot have trust in the blockchain system. If a malicious attacker could manage to attack this atomic ordering process, then disaster could happen. One of the typical attacks is "double spending". This kind of attack has already happened in blockchains such as Zencash, Ethereum Classic, and Bitcoin Gold. Generally, a secure blockchain should satisfy the following two conditions:

1. Safety: honest participants should agree on the order of valid transactions and no forks should happen.

2. Liveness: a valid transaction should be included in the blockchain in a reasonable time period.

For a PoS based blockchain, participants satisfying certain conditions could be invited to generate a candidate next block. Normally there are a few participants that satisfy these conditions. Since it is profitable for a participant to generate blocks in the blockchain, each participant would try his/her best to convince the community to accept his/her block as the next block. If two different blocks are accepted as the next block by the community at the same time, a fork is created and the safety property is no longer satisfied. In order to prevent fork from happening, most PoS based blockchains use Byzantine Fault Tolerance (BFT) protocols as a finality gadget to select a unique next-block from a few candidate next-blocks. There are a few challenges for this process to work correctly.

2.1 Challenge 1: Existing BFT protocols are not sufficient to guarantee blockchain security in open networks

As we have mentioned in the preceding paragraphs, normally there are a few participants who are qualified to produce the next candidate block. BFT consensus protocols are frequently used as a finality gadget to select a unique next-block from a few candidate next-blocks. However, a recent analysis by Yongge Wang [37] shows that several widely deployed BFT based PoS blockchains may be insecure.

Lamport, Shostak, and Pease [25] and Pease, Shostak, and Lamport [29] initiated the study of reaching consensus in face of Byzantine failures and designed the first synchronous solution for Byzantine agreement. Dolev and Strong [14] proposed an improved protocol in a synchronous network with $O(n^3)$ communication complexity. By assuming the existence of digital signature schemes and a public-key infrastructure, Katz and Koo [23] proposed an expected constant-round BFT protocol in a synchronous network setting against $\lfloor \frac{n-1}{2} \rfloor$ Byzantine faults.

For an asynchronous network, Fischer, Lynch, and Paterson [17] showed that there is no deterministic protocol for the BFT problem in face of a single failure. Several researchers have tried to design BFT consensus protocols to circumvent the impossibility. For example, Ben-Or [3] initiated the probabilistic approach to BFT consensus protocols in completely asynchronous networks and Dwork, Lynch, and Stockmeyer [15] designed BFT consensus protocols in partial synchronous networks. Castro and Liskov [7] initiated the study of practical BFT consensus protocol design and introduced the PBFT protocol for partial synchronous networks. The core idea of PBFT has been used in the design of several widely adopted BFT systems such as Tendermint BFT [6]. Tendermint has been used in more than 40% deployed Proof of State blockchains (see, e.g., [24]) such as the "Internet of Blockchain" Cosmos [11]. More recently, Yin et al [38] improved the PBFT/Tendermint protocol by changing the mesh communication network in PBFT to star communication networks in HotStuff and by using threshold cryptography. Facebook's Libra blockchain has adopted HotStuff in their LibraBFT protocol [32].

In the literature, there are mainly two kinds of partial synchronous networks for Byzantine Agreement protocols. In Type I partial synchronous networks, all messages are guaranteed to be delivered. In this type of networks, Denial of Service (DoS) attacks are not allowed and reliable point to point communication channels for all pairs of participants are required for the underlying networks. In Type II partial synchronous networks, the network becomes synchronous after an unknown Global Synchronization Time (GST). In this type of networks, Denial of Service (DoS) attacks are allowed before GST though it is not allowed after GST. The Type II network is more realistic and is commonly used in the literature. For example, the partial synchronous network model in LibraBFT [32] is defined as "*the network alternates between periods of bad and good connectivity, known as periods of asynchrony and synchrony, respectively...During periods of asynchrony, we allow messages to be lost or to take an arbitrarily long time. We also allow honest nodes to crash and restart*". Our Internet environments can be modeled as a Type II partial synchronous network.

Many BFT protocols in partial synchronous networks use reliable broadcast protocols for certain message transmission. In particular, these protocols normally leverage the gossip-based broadcast protocol in Bracha [5] which is based on the existence of reliable point-to-point communication channels for all pairs of participants. In particular, the broadcast protocol in Bracha [5] assumes a complete network to achieve "*a reliable message system in which no messages are lost or generated.*". Since our Internet infrastructure is not a complete network, one needs to be very careful in building Internet based BFT protocols using Bracha's results. Specifically, one should not assume that there is a reliable broadcast channel before GST of Type II networks.

The recent paper by Yongge Wang [37] contains a security analysis for several PoS based blockchains that are based on the following BFT consensus protocols: Tendermint BFT, Polkadot's GRANDPA BFT, Ethereum's Casper the Friendly Finality Gadget, and Hotstuff (also Facebooks' Libra BFT which is based on Hotstuff). In next Sections, we will present SperaxBFT consensus protocol whose security is mathematically provable in asynchronous networks (our Internet is normally considered as an asynchronous network). For more detailed in secure BFT protocols in partially synchronous networks and on the mathematical proofs for SperaxBFT protocols, the reader is referred to Wang [37].

2.2 Challenge 2: achieving unpredictability of next-block-proposer identity

It is very important to guarantee unpredictability of the participant identity who would produce the next candidate block. If the next candidate block producer identity is predictable, then there are many potential attacks. As an example, we can describe two simple scenarios. In the first scenario, a malicious user may bribe the next candidate block producer to exclude certain transactions within the next block. In the second scenario, assuming that both A and B are the next candidate block producers. If A knows the identity of B, A may try to DoS B's network so that B's candidate block could not be broadcast to the network. Thus A's candidate block will be accepted as the final next-block. Several attacks based on the predictability of next-block producer identity has happened already. For example, this has happened several times on the EOSIO Blockchain Software architecture. Many currently deployed PoS blockchains do not address this challenge at all. Several other blockchains leverage the Verifiable Random Function (VRF) techniques to address this challenge. In next next section, we will describe Sperax's innovative random beacon protocol to guarantee the unpredictability of the next-block producer identity.

3 Sperax's Innovative layer-1 design

In this section, we present Sperax's innovative layer one design that address the major challenges that we have discussed in the preceding section.

3.1 Sperax blockchain

Sperax proposes a consensus mechanism to simultaneously achieve decentralization, performance and security. 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. 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

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

Sperax 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 Sperax, 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 Sperax, we assume that the network is a Type II partial synchronous networks. Specifically, we assume that, at the start of round r, most honest 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 $\kappa = 10$. The choice of $\kappa > 0$ will guarantee that the system is running in a consistent way and also will defeat adaptive key selection attacks (where the adversary spawns new participants with appropriately chosen public keys to increase the chance that these newly spawned participants have a high probability to be chosen as the block proposer).

3.2 Sperax random beacon protocol

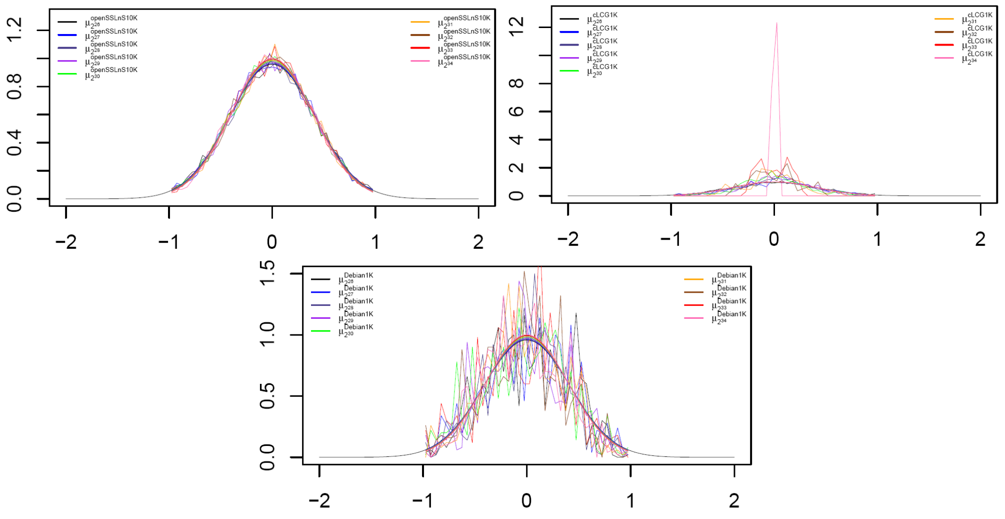
In Sperax, a participant is selected to serve as the candidate next-block proposer or as a BFT committee member based on a few random entropy. In a high level, the entropy is from three sources: a random string that the participant would commit in a previous round, randomness from the blockchain itself, and a random string that is based on a coin-flipping protocol such as Randao. The first random source guarantees that the participant identity is not revealed until the participant publishes his/her proposed next candidate block. The third random source guarantees that the process is decentralized, multi-participatory, conspiracy-resistant, and verifiably fair.

Random numbers have been one of the most useful objects in statistics, computer science, cryptography, modeling, simulation, and other applications though it is very difficult to construct true randomness. Many solutions (e.g., cryptographic pseudorandom generators) have been proposed to harness or simulate randomness and many statistical testing techniques have been proposed to determine whether a pseudorandom generator produces high quality random bits.

A string is said to be cryptographically pseudorandom if no efficient observer can distinguish it from a uniformly chosen string of the same length. There have been extensive efforts in finding approaches to generate true random strings. For example, the radioactive decay based random bit generator [33], atmospheric noise (e.g., from radio stations) based random bit generators [30], optical ultrafast random bit generator [22], thermal noise based random bit generator in Intel processors [21], optoelectronic random bit generators based on laser chaos [26], and other system environment based random bit generators (e.g., random bits based on NIC, mouse cursor positions, user inputs, etc.) have been extensively used in practice.

The weakness in pseudorandom generators could be used to mount a variety of attacks on blockchain or Internet security. It is reported in Debian Security Advisory DSA-1571-1 [12] that the random number generator in Debian's OpenSSL release CVE-2008-0166 is predictable. The weakness in Debian pseudorandom generator affected the security of OpenSSH, Apache (mod_sl), the onion router (TOR), OpenVPN, and other applications (see, e.g., [1]). These examples show that it is important to implement high quality pseudorandom generators and to implement real-time randomness monitoring tools for blockchains. Although there are state-of-the-art pseudorandom testing techniques such as NIST SP800-22, Yongge Wang and Tony Nicol [36] have carried out extensive experiments (based on over 200TB of random bits generated) and showed that NIST SP800-22 techniques could not detect statistical weakness in several widely deployed pseudorandom generators such as the weak Debian pseudorandom generator. Wang and Nicol [36] then proposed a "behavioristic" testing approach to address these challenges. In particular, they designed Brownian motion and the law of the iterated logarithm based testing techniques that could be used to detect statistical weakness that NIST testing suite is insufficient to detect. Sperax blockchain monitoring interface will integrate Wang and Nicol's Brownian motion testing technique [36] to show the real-time statistical (Brownian motion/LIL test) properties of various random sources employed in Seprax blockchain. When certain attacks on the randomness entropy are mounted, the testing interface should be able to display this and then the community may take appropriate actions together. Figure 1 is an example interface that shows the statistical weakness of certain pseudorandom generators. Specifically, Figure 1 shows that the statistical distributions of the following pseudorandom generators: the first image is the Brownian motion (LIL test) output for the secure randomness generator, the second image is the output for Linear congruential generators (LCG), the third image is the output for the weak Debian Linux generator. It is straightforward to observe from these graphs that the LCG and Debian Linux pseudorandom generators are insecure. In the Figure, the LCG generator is based on Microsoft Visual Studio stdlib function rand() which uses the standard C linear congruential generator (each sequence is generated with a 4-byte seed from random.org [30].

Figure 1: LIL test output for Secure OpenSSL generators, LCG generators, and weak Debian generators



Sperax's pseudorandom generator leverages the commit-reveal and BLS schemes proposed in the randao protocol [16]. However, the randao protocol is only a public coin-flip protocol and is not sufficient to protect the candidate next-block proposer's identity before the candidate block is published. Sperax achieves the block proposer' identity protection via a further commit-reveal process. Sperax 1.0 will implement the commit-reveal protocol for the common coin while Sperax 2.0 will implement BLS based protocol as specified in [16] for the common coin. In the commit-reveal based common coin, the participant is required to publish his/her random string after the commitment, if he/she does not publish his/her committed string, his/her stake would be slashed. However, if a single participant chooses not to post the committed string, the common coin production process is considered failed and a new commitment process is required. Thus a malicious adversary may try to attack the system by sacrificing his/her tokens. This challenge will be addressed by using the BLS based common coin protocol in Sperax 2.0 since the BLS threshold signatures are generated by a group of people, no individual can predict the results, and a single participant cannot stop the post of signature.

For Sperax, if a participant P_i wants to participate in some rounds between $r+\kappa'+1$ and $r+\kappa'+\kappa''$ (where $\kappa'<\kappa$) as a potential block-proposer or as a potential verifier, P_i chooses a random string γ_i and computes $\kappa''+1$ random sequences $\gamma_i^{r+\kappa'} = H(\gamma_i^{r+\kappa'+1}), \gamma_i^{r+\kappa'+1+1} = H(\gamma_i^{r+\kappa'+2}), \cdots, \gamma_i^{r+\kappa'+\kappa''} = H(\gamma_i)$. Then the participant P_i submits a transaction

$$SIG_{pk_i}\left(P_i, P_\infty, m_i, \gamma_i^{r+\kappa'}, r+\kappa'\right)$$

to be included in block B^r, where P_∞ is a virtual users for accepting random sequence commitment and $m_i = 10,000a_i$ is the amount of tokens to be staked ($a_i \geq 1$ is an integer).

In Sperax 2.0, the common coin $coin_{r+\kappa'}$ for rounds from $r+\kappa'+1$ to $r+\kappa'+\kappa''$ will be derived using the randao BLS common coin protocol. The details for the randao BLS common coin protocol is referred to [16].

3.3 Reliable broadcast and BFT protocols

Assume that the time is divided into discrete units called slots $T_0, T_1, T_2, \cdots$ where the length of the time slots are equal. Furthermore, we assume that: (1) the current time slot is determined by a publicly-known and monotonically increasing function of current time; and (2) each participant has access to the current time. In a synchronous network, if an honest participant P_1 sends a message m to a participant P_2 at the start of time slot T_i, the message m is guaranteed to arrive at P_2 at the end of time slot T_i. In the complete asynchronous network, the adversary can selectively delay, drop, or re-order any messages sent by honest parties. In other words, if an honest participant P_1 sends a message m to a participant P_2 at the start of time slot T_{i_1}, P_2 may never receive the message m or will receive the message m

eventually at time T_{i_2} where $i_2 = i_1 + \Delta$. Dwork, Lynch, and Stockmeyer [15] considered the following two kinds of partial synchronous networks:

- Type I asynchronous network: $\Delta < \infty$ is unknown. That is, there exists a Δ but the participants do not know the exact value of Δ.
- Type II asynchronous network: $\Delta < \infty$ holds eventually. That is, the participant knows the value of Δ. But this Δ only holds after an unknown time slot $T = T_i$. Such a time T is called the Global Stabilization Time (GST).

For Type I asynchronous networks, the protocol designer supplies the consensus protocol first, then the adversary chooses her Δ. For Type II asynchronous networks, the adversary picks the Δ and the protocol designer (knowing Δ) supplies the consensus protocol, then the adversary chooses the GST. The definition of partial synchronous networks in [7, 38, 32] is the second type of partial synchronous networks. That is, the value of Δ is known but the value of GST is unknown. In such kind of networks, the adversary can selectively delay, drop, or re-order any messages sent by honest participants before an unknown time GST. But the network will become synchronous after GST.

For the Type I network model, Denial of Service (DoS) attack is not allowed since message could be lost with DoS attacks. We think that it is more natural to use Type II asynchronous networks for distributed BFT protocol design and analysis. Sperax adopts the Type II network model unless specified otherwise.

The difference between point-to-point communication channels and broadcast communication channels has been extensively studied in the literature. A reliable broadcast channel requires that the following two properties be satisfied.

1. Correctness: If an honest participant broadcasts a message m, then every honest participant accepts m.
2. Unforgeability: If an honest participant does not broadcast a message m, then no honest participant accepts m.

Other broadcast primitives have also been proposed in the literature (see, e.g., Mullender [27]):

1. FIFO broadcast: a reliable broadcast guaranteeing that messages broadcast by the same honest sender are delivered in the order they were broadcast.
2. Causal broadcast: a reliable broadcast guaranteeing that messages are delivered according to the causal precedence relationship. That is, if a message m depends on m' then m' is delivered before m.
3. Atomic broadcast: a reliable broadcast guaranteeing a total ordering of all messages. Unlike causal broadcasts, atomic broadcast requires all participants to receive all messages in the same order. However, the atomic broadcast does not enforce a causal order.
4. FIFO atomic broadcast: FIFO broadcast a total ordering of all messages.
5. Causal atomic broadcast: casual broadcast a total ordering of all messages.

It has been shown in Chandra and Toueg [8] that atomic broadcast is equivalent to Byzantine consensus. Other related group broadcast primitives for communication for supporting distributed computation in the presence of non-Byzantine crash failures may be found in Birman and Joseph [4] and other literatures.

For complete networks, reliable broadcast protocols have been proposed in Bracha [5]. For a given integer k, a network is called k-connected if there exist k-node disjoint paths between any two nodes within the network. In non-complete networks, it is well known that $(2t + 1)$-connectivity is necessary for reliable communication against t Byzantine faults (see, e.g., Wang and Desmedt [35] and Desmedt-Wang-Burmester [13]). On the other hand, for broadcast communication channels, Wang and Desmedt [34] showed that there exists an efficient protocol to achieve probabilistically reliable and perfectly private communication against t Byzantine faults when the underlying communication network is $(t + 1)$-connected. The crucial point to achieve these results is that: in a point-to-point channel, a malicious participant P_1 can send a message m_1 to participant P_2 and send a different message m_2 to participant P_3 though, in a broadcast channel, the malicious participant P_1 has to send the same message m to multiple participants including P_2 and P_3. If a malicious P_1 sends different messages to different participants in a reliable broadcast channel, it will be observed by all receivers.

Though broadcast channels at physical layers are commonly used in local area networks, it is not trivial to design reliable broadcast channels over the Internet infrastructure since the Internet connectivity is not a complete graph and some direct communication paths between participants are missing (see, e.g., [25, 35]). Quite a few broadcast primitives have been proposed in the literature using message relays (see, e.g., Srikanth and Toueg [31], Bracha [5], Dwork, Lynch, and Stockmeyer [15], and LibraBFT [32]). In the message relay based broadcast protocol, if an honest participant accepts a message signed by another participant, it relays the signed message to other participants. However, in order for these message relay based broadcast protocol to be reliable, it requires that the network graph is complete which is not true for the Internet environments.

A broadcast channel is *unreliable* if a malicious participant could broadcast a message m_1 to a proper subset of the participants but not to other participants. That is, some participants will receive the message m_1 while other participants will receive a different message m_2 or receive nothing at all. In next sections, we show that several BFT protocols are insecure due to the lack of reliable broadcast channels before GST (messages before GST could get lost or re-ordered by the definition). Thus it is important to design BFT protocols that could tolerate unreliable broadcast channels before GST.

Sperax BFT protocol. Sperax adopts a Type II partial synchronous network model and adopts the BDLS BFT protocol discussed and analyzed in Wang [37] as the finality gadget for the Sperax block chain. For details of the BDLS BFT protocol and analysis, it is referred to [37].

3.4 Mathematically Provable Secure BFT Consensus Protocol: SperaxBFT

As we have mentioned in Section 3.2, if a user P_i wants to participant in the leader and verifier committee selection process for the block B^r generation, then P_i needs to deposit $10000a_i$ SPA tokens in round $r-\kappa'$. Note that a user may create several virtual participants for the governance process of Sperax blockchain and each virtual participant deposits 10000 SPA tokens. Alternatively, a user may commit $10000a_i$ SPA tokens in one package to improve the performance. Let Ω_r be the total amount of SPA tokens deposited for the round r. For each round r, the protocol proceeds from the stage $s=0$ until a block B^r is finalized. If stage s fails to finalizes the block B^r, it moves to stage $s+1$.

Specifically, a stage s consists of two steps. The potential leader is selected during the first step and the verifier committee is selected during the second step. A potential leader of stage s of round r is a user P_i if and

$$\frac{H(r,0,\gamma_i^r,coin_r,W_r)}{2^{256}-1} > \max\{0, 1-a_ip\} \tag{1}$$

where γ_i^r is the random string that user P_i has committed with staking (see Section 3.2), and W_r is defined as follows:

1. $W_0 = H(\texttt{0x03243F6A8885A308D313198A2E037073})$,
2. $W_j = H(P^{j-1}, W_{j-1})$ for $0 < j \le r$, where $P^0=\emptyset$ and P^j is the participant who generated the block B^{j-1},

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. For example, one may choose $p=\frac{5}{T_r}$ where T_r is the total number of unit users who paid staking deposit for the round r (that is, $T_r=\Omega_r/10000$). Since the user P_i cannot modify the value $R_{i,r,2s}$ after she/he has submitted the commitment in an early round, it is guaranteed that a user P_i cannot increase her probability to be a leader by changing the value $R_{i,r,2s}$. 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 μ_i^r that she committed in an early round. Other users can verify that the user P_i is a potential leader after P_i discloses the committed string μ_i^r. The leader P_{l^r} is defined to be the user whose hashed random string is the largest. That is,

$$\frac{H(r,0,\gamma_{l_r}^r,coin_r,W_r)}{2^{256}-1} \ge \frac{H(r,0,\gamma_i^r,coin_r,W_r)}{2^{256}-1} \tag{2}$$

for all potential leaders P_i.

Remark: It should be noted that if a user stakes $10000a_i$ SPA tokens and uses (1) to calculate the probability for him to serve as the block B^r proposer, he may has small disadvantages since he only has one chance to use the equation (2). If this is a concern, for a user who staked $10000a_i$ SPA tokens, he may be considered as a_i virtual users for the calculation of the probability in the equation (1).

At the start of stage s of round r, each potential leader P_i of this stage 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 to the entire network the message $m_i^{r,2s} = (B_i^r, \mu_i^r)$ together with the Merkle hash tree path corresponding the value μ_i^r.

Since there could be several potential leaders during round r, each user could receive several candidate block messages $m_j^{r,2s}$ 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 $\tau_i > 0$ times (that is, P_i's signature will be counted as τ_i signatures) where

$$\tau_i = \min_{\tau}\left\{\frac{H(r, 1, \gamma_i^r, coin_r, W_r)}{2^{256}-1} < \sum_{j=0}^{\tau}\binom{a_i}{j}(p')^j(1-p')^{a_i-j}\right\} \tag{3}$$

where $\gamma_i^{r'}$ is the committed string by user P_i in Section 3.2, 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 $p' = \frac{60}{T_r}$ where T_r is the total number of unit users who paid staking deposit for the round r (that is, $T_r = \Omega_r/10000$). 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 $\frac{p'}{\Omega_r}$ (this approach is similar to the approach used in Gilad et al [19]).

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_j contained in the messages $m_j^{r,2s}$ that she has received. After verifying the validity of the message $m_l^{r,2s} = (B_l^r, \mu_l^r)$, 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. Together with the message, the user P_i should also propagates the proof that she is the qualified committee member. This can be done by including his committed value μ_i^r together with the Merkle hash tree path corresponding the value μ_i^r.

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 like to work voluntarily to keep the system in function.

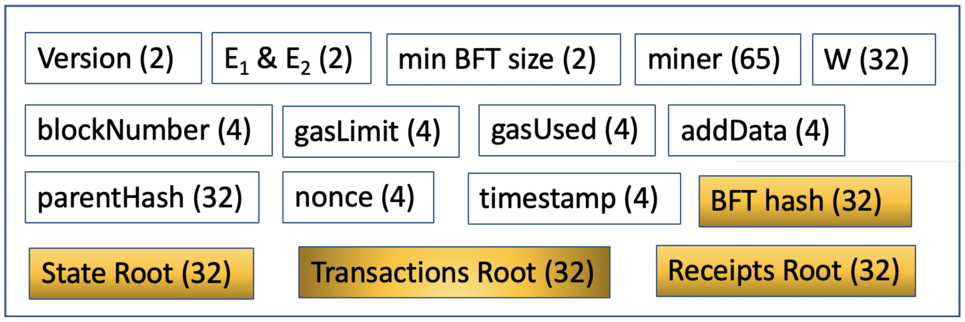
3.5 Sperax blockchain header

One block header for Sperax block consists of 287 bytes. The Figure 2 shows the header of a Sperax block.

3.6 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 specified 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.

Figure 2: Sperax Blockchain Header



3.6.1 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 networks always know the *actual activity states* of all nodes. (Even if some geeky malicious nodes have succeeded in cheating on the Sperax in terms of their activeness, the Sperax nework 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, we consider our network broken and not safe anymore.

Factors. In a nutshell, the probability for malicious nodes to successfully make the first block when attempting to make a fork, $Prob_r$, 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 $Prob_r$, we must first understand the two essential steps in which malicious nodes have to succeed if they ever want to successfully make their 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 probabilistically independent, the ultimate probability for malicious nodes to successfully make the first block in forming a fork, $Prob_r$, is simply the multiplication of the two. That is,

$$Prob_r = Prob_r^1 * Prob_r^2 \tag{4}$$

Now, in the following, we explain how P_r^1 and P_r^2 are calculated respectively. We calculate $Prob_r^1$ by the formula below:

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

where M is the amount of money controlled by all 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 Sperax network-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). Since $(1-p_r)^M$ is the probability of no malicious node elected as a potential leader, $1 - (1 - p_r)^M$ means the probability of they having at least one 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 Sperax network-chosen probability parameter as discussed in section 3 ($p'_r = \frac{N'_r}{T_r}$); V_{min} is also a Sperax network-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.

3.6.2 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 list of intermediate parameters involved when calculating $Prob_r$ by our formula:

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

Calculate $Prob_r^1$:

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

Calculate $Prob_r^2$:

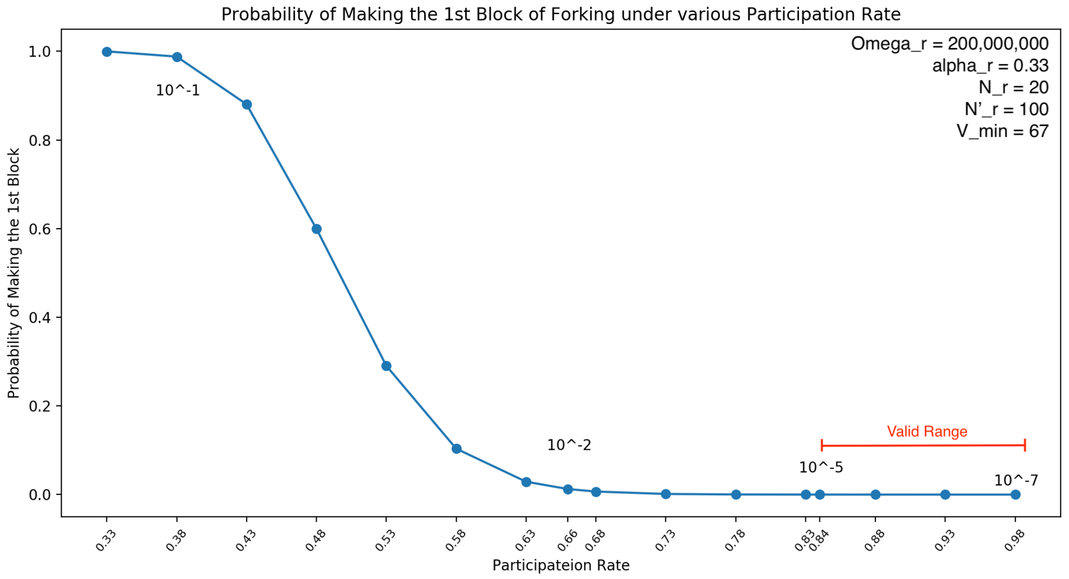
$$Prob_r^2 = 1 - \sum_{i=0}^{66} \binom{66,000,000}{i} \left(\frac{1}{1,680,000}\right)^i \left(1 - \frac{1}{1,680,000}\right)^{66,000,000-i} \approx 3.5827e - 05$$

Finally, calculate $Prob_r$:

$$Prob_r = Prob_r^1 * Prob_r^2 \approx 0.9996130 * 3.5827e - 05 \approx \textbf{3.5813e-05}$$

3.6.3 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 Sperax network-controlled parameters. In orther words, the Sperax network 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 α_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 Sperax network (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 Sperax network-controlled parameters are appropriately chosen.

3.7 Democratic and Decentralized

Sperax aims to build a fair, just and open community with self-governance, self-motivation and sustainable development. Sperax gives each user a fair chance to become a potential leader.

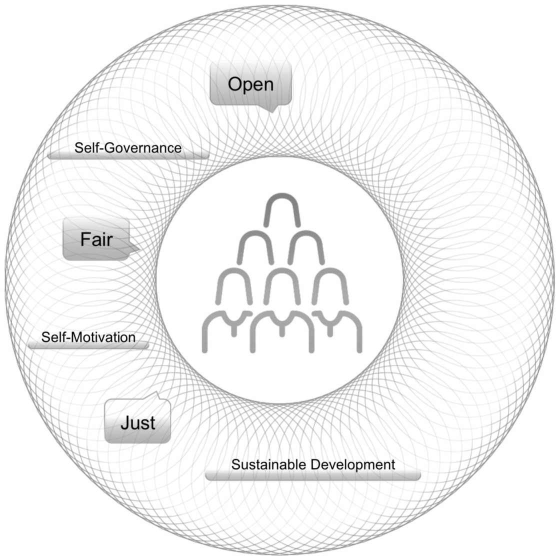


Figure 3: Democratic community

Sperax realizes the privacy, integrity, authenticity and reliability of information in an open environment of distrust through the comprehensive use of multiple security technologies. The random numbers generated by Sperax random beacon protocol is used to select the potential leaders and validation committee members. A secure and efficient BFT protocol is used by the validation committee to select the leader from the potential leaders. There is no user in Sperax playing the role of a super user/node with the ability to gain unearned rewards through their monopoly power and position. Sperax is expected to be a complete decentralized community, which is similar to Bitcoin.

3.8 Virtual Machine and Various Applications

Bitcoin's Forth-like scripting language has limited capability. In order to support more powerful smart contracts, Ethereum adopted Turing complete scripting languages. Since the halting problem for universal Turing machine is undecidable, this introduces serious challenges for determining the safety and correctness of smart contracts. Without a validation mechanism for certain important applications, attacks could be easily mounted on the blockchain. The notorious example is the DAO attack [20] against Ethereum that has happened in 2016. To make things worse, there is no effective approach to avoid such kind of attacks in future.

Sperax takes an alternative approach to address this challenge. Our approach is similar to the Microsoft Static Driver Verifier (SDV) [2]. Sperax provides a restricted non-Turing-complete scripting language that could be used to develop Sperax smart contracts. Sperax smart contracts are executed on Sperax Virtual Machines (SVM). The non-Turing completeness of Sperax scripting language provides the feasibility of developing automatic verification mechanisms to verify the correctness of Sperax smart contracts. Specifically, using ideas from symbolic model checking, program analysis and theorem proving, the verification engine automatically checks that a Sperax smart contract correctly uses the interface to an external library. Furthermore, the verification engine can systematically analyze the smart contract source code against a set of rules that define what it means for a smart contract to properly interact with the Sperax Virtual Machines (SVM). The automatic verification system provides a novel method to support various applications including finance, advertising, insurance, media, games, and other potential industries. The Sperax team has developed supporting modules that help developers to conveniently deploy their applications over Sperax blockchain.

4 Sperax Tech Roadmap

4.1 Phase 1: Prototype

- Feasibility Research
  1. Focus on SperaxBFT consensus mechanism and prove its validity;
  2. Sperax Basic Model Prepare;
  3. Privacy transaction technique research;
  4. Put up with privacy transaction Proposal/Design in Sperax.
- Software design
  1. Complete detailed design of software;
  2. Focus on the designing of function, performance, input and output, algorithms, logic, interface, storage distribution.
- Software Development & Test
  1. SperaxBFT consensus algorithm verified in software environment and prompted;
  2. Privacy transaction performance tested in software environment and optimization;
  3. Sperax blockchain runs in complete software environment and achieves all functions which are mentioned in White Paper;
  4. Provide Sperax blockchain traceability;

4.2 Phase 2: Full version

- Production & Test
  1. Sperax testnet Wallet release;
  2. Sperax testnet online.
- Release
  1. Release Sperax (Alpha) ;
  2. Release SperaxScan (SperaxScan support entire Sperax blockchain block include basic transaction and smart contract transaction);
  3. Release Sperax (Belta) ;
  4. Sperax MainNet Wallet release;
  5. Sperax main-net online (v1.0).
- Future plan
  1. Improve SperaxVM to achieve higher performance and security.

References

[1] David Ahmad. Two years of broken crypto: debian's dress rehearsal for a global pki compromise. *Security & Privacy, IEEE*, 6(5):70–73, 2008.

[2] Thomas Ball, Byron Cook, Vladimir Levin, and Sriram K Rajamani. SLAM and static driver verifier: Technology transfer of formal methods inside microsoft. In *International Conference on Integrated Formal Methods*, pages 1–20. Springer, 2004.

[3] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proc. 2nd ACM PODC*, pages 27–30, 1983.

[4] K.P. Birman and T.A. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems (TOCS)*, 5(1):47–76, 1987.

[5] G. Bracha. An asynchronous $[(n-1)/3]$-resilient consensus protocol. In *Proc. 3rd ACM PODC*, pages 154–162. ACM, 1984.

[6] E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on BFT consensus. *Preprint arXiv:1807.04938*, 2018.

[7] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM TOCS*, 20(4):398–461, 2002.

[8] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *JACM*, 43(2):225–267, 1996.

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

[10] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proc. CRYPTO*, pages 319–327. Springer-Verlag New York, Inc., 1990.

[11] Cosmos. Cosmos Network: Internet of Blockchains `https://cosmos.network`.

[12] Debian. Debian security advisory dsa-1571-1. available at `http://www.debian.org/security/2008/dsa-1571.`, 2008.

[13] Yvo Desmedt, Yongge Wang, and Mike Burmester. A complete characterization of tolerable adversary structures for secure point-to-point transmissions without feedback. In *International Symposium on Algorithms and Computation*, pages 277–287. Springer, 2005.

[14] D. Dolev and H.R. Strong. Polynomial algorithms for multiple processor agreement. In *Proc. 14th ACM STOC*, pages 401–407. ACM, 1982.

[15] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *JACM*, 35(2):288–323, 1988.

[16] Ethereum. Randao: A DAO working as RNG of ethereum, 2017. `https://github.com/randao/randao` and `https://www.randao.org/whitepaper/Randao_v0.85_en.pdf`.

[17] M.J. Fischer, N. A Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.

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

[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proc. the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

[20] Osman Gazi Gucluturk. The dao hack explained: Unfortunate take-off of smart contracts. `https://medium.com/@ogucluturk/the-dao-hack-explained-unfortunate-take-off-of-smart-contracts-2bd8c8db3562`.

[21] Benjamin Jun and Paul Kocher. The intel random number generator. *Cryptography Research Inc. white paper*, 1999.

[22] I. Kanter, Y. Aviad, I. Reidler, E. Cohen, and M. Rosenbluh. An optical ultrafast random bit generator. *Nature Photonics*, 4(1):58–61, 2010.

[23] J. Katz and C.-Y. Koo. On expected constant-round protocols for byzantine agreement. *Journal of Computer and System Sciences*, 75(2):91–112, 2009.

[24] J. Kwon. Tendermint powers 40%+ of all proof-of-stake blockchains. invest:asia, available at `https://realsatoshi.net/12886/`, Sept. 12, 2019.

[25] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[26] Pu Li, Xiaogang Yi, Xianglian Liu, Yuncai Wang, and Yongge Wang. Brownian motion properties of optoelectronic random bit generators based on laser chaos. *Optics express*, 24(14):15822–15833, 2016.

[27] S.J. Mullender. *Distributed systems*. Addison-Wesley, 1993.

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

[29] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *JACM*, 27(2):228–234, 1980.

[30] RANDOM.ORG. Random.org `http://www.random.org/`.

[31] TK Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.

[32] The LibraBFT Team. State machine replication in the Libra Blockchain. available at `https://developers.libra.org/docs/assets/papers/libra-consensus-state-machine-replication-in-the-libra-blockchain/2019-11-08.pdf`, November 28, 2019.

[33] John Walker. Hotbits: Genuine random numbers, generated by radioactive decay. *Online: http://www.fourmilab.ch/hotbits*, 2001.

[34] Y. Wang and Y. Desmedt. Secure communication in multicast channels: the answer to Franklin and Wright's question. *Journal of Cryptology*, 14(2):121–135, 2001.

[35] Y. Wang and Y. Desmedt. Perfectly secure message transmission revisited. *Information Theory, IEEE Tran.*, 54(6):2582–2595, 2008.

[36] Y. Wang and T. Nicol. Statistical properties of pseudo random sequences and experiments with PHP and Debian OpenSSL. In *Proc. ESORICS, LNCS 8712*, pages 454–471, 2014.

[37] Yongge Wang. Byzantine fault tolerance in partially connected asynchronous networks. *`http://eprint.iacr.org/2019/1460`*, 2019.

[38] M. Yin, D. Malkhi, M.K. Reiter, G.G. Gueta, and I. Abraham. HotStuff: BFT consensus in the lens of blockchain. *arXiv preprint arXiv:1803.05069*, 2018.

A Sperax Enabling Technology

A.1 Privacy Protection Technology

Sperax leverages several cryptographic algorithms to provide reliable and secure privacy protection. For some applications, Sperax uses efficient additive homomorphic encryption schemes. For more involved applications, Sperax supports fully homomorphic encryption mechanisms. Sperax also developed a SperaxDAP System. The Sperax Decentralized Anonymous Payment system (SperaxDAP) combines the idea of coin-join and transaction encryption to achieve an anonymous payment which protects the privacy of payment address, amount and receiver address. The SperaxDAP uses "Pedersen Commitment" homomorphism from the BGN06 encryption scheme to verify the transaction amount and enable the transaction payee to decrypt the ciphertext directly without additional communication. In addition, SperaxDAP adopts the neoteric NIZK certification technology named BulletProofs to verify the transaction value interval as well as the BGL03 aggregated signature (OWAS) and the coin-join technology to protect users' identity privacy. By deleting the independent transaction signatures and the public keys of the encryption scheme, the number of transactions stored in the block is increased. An example of SperaxDAP is presented in Figure 4.

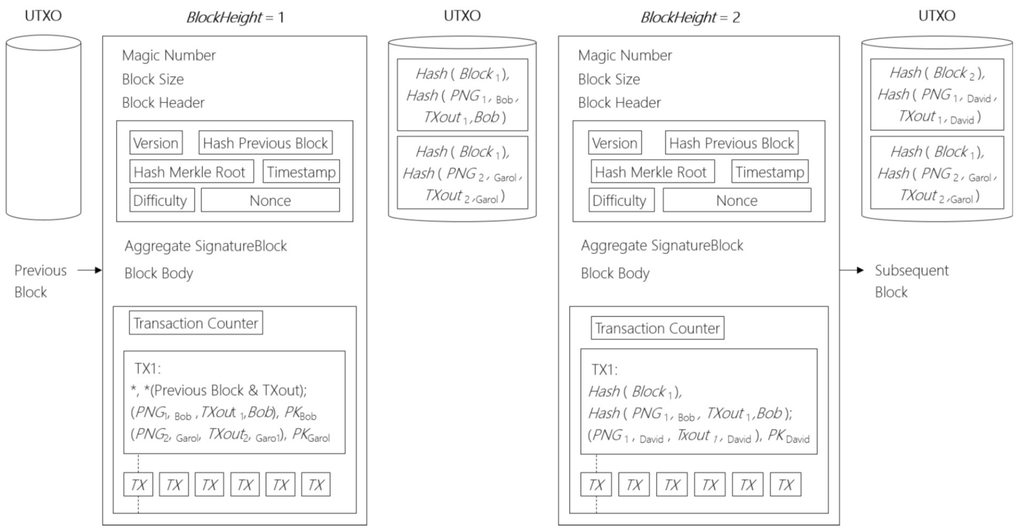

UTXO
BlockHeight = 1
UTXO
BlockHeight = 2
UTXO
Magic Number
Block Size
Block Header
Version
Hash Previous Block
Hash Merkle Root
Timestamp
Difficulty
Nonce
Aggregate SignatureBlock
Block Body
Transaction Counter
TX1:
*, *(Previous Block & TXout);
(PNG1, Bob, TXout 1, Bob), PKBob
(PNG2, Garol, TXout2, Garol), PKGarol
TX
Hash (Block 1), Hash (PNG 1, Bob, TXout 1, Bob)
Hash (Block 1), Hash (PNG 2, Garol, TXout 2, Garol)
Hash (Block 2), Hash (PNG 1, David, TXout 1, David)
TX1:
Hash (Block 1),
Hash (PNG 1, Bob, TXout 1, Bob);
(PNG 1, David, Txout 1, David), PK David
Previous Block
Subsequent Block

Figure 4: SperaxDAP example

A.2 Smart Contract

Contract code storage: The hash value of the smart contract is recorded on the chain, and the complete contract code is stored off chain with a hash value index. When the contract is executed, the code is loaded from the outside of chain. Since the hash value of the contract is already recorded in the chain, even if the code is loaded from the outside of chain, there is no need to worry about the contents of the contract being tampered with. This allows the node to save more space for storage and can also protect the content of smart contracts to a certain degree of privacy at the same time.

Contract code execution: The Sperax virtual machine adopts a memory-safe and platform-independent instruction set standard based on WebAssembly that can be mapped to all types of CPU architectures perfectly and efficiently. The contract's instruction set runs in a virtual container which imposes strict restrictions on computing resources and program steps to ensure secure execution of the contract.