SWIFT BLOCK

ADVANCED NETWORK

SWIFTBLOCK WHITEPAPER

目录

Abstract

SwiftBlock Coin is a revolutionary cryptocurrency focused on traffic tokenization. Through decentralized traffic tokenization, SwiftBlock Coin provides a transparent, secure, and efficient transaction method. Built on the SwiftBlock Consensus Protocol (SCP), SwiftBlock Coin adopts a Decentralized Autonomous Organization (DAO) framework, ensuring fair governance and

community-driven development. A unique referral reward mechanism promotes user growth, incentivizes reward distribution, and enhances ecosystem activity.

1. Introduction

1.1. Introduction to Traffic Tokenization

SwiftBlock Coin is a revolutionary cryptocurrency focused on traffic tokenization, breaking away from traditional traffic monetization methods. By integrating traffic tokenization with blockchain technology, it creates new channels for traffic monetization.

1.2. Current Status and Future Expectations of the Traffic Monetization Market

According to the latest market research data, the global advertising industry's market value surpassed the $1 trillion mark for the first time in 2024. Digital advertising dominates this space, expected to account for 73% of global advertising expenditures by 2025. In the Chinese market, the digital advertising market reached billion RMB in 2024 and is projected to grow at a compound annual growth rate (CAGR) of over 15% by 2029.

1.2.1. Currently, the global internet traffic monetization market primarily generates revenue through advertising, membership subscriptions, and data transactions. Due to its broad scope and involvement across multiple industries, it is challenging to accurately estimate its total market value.

1.2.1.1. China's Influencer Economy Market: According to Xinhua News, China's influencer economy market reached RMB 1.3 trillion in 2022, a year-on-year increase of 26.9%. It is expected to exceed RMB 1.6 trillion in 2023, with a year-on-year growth of 23.8%.

1.2.1.2. China's Short Video and Live Streaming Market: According to the "2023 National Radio and Television Industry Statistical Bulletin," the revenue from short videos, live streaming, and other online audiovisual-related businesses in China exceeded RMB 420 billion in 2022, a year-on-year increase of 33.39%.

1.2.2. Looking ahead, with the development of blockchain, artificial intelligence, and Web3.0 technologies, the traffic monetization market is expected to witness new growth opportunities. Decentralized traffic trading, enhanced privacy protection, and the application of smart contracts will further expand the market size.

1.3. Current Issues in the Traffic Monetization Market

1.3.1. **Market Dominance by Large Companies (Monopoly)** – Internet traffic monetization is primarily controlled by tech giants such as Google, Meta

(Facebook), Alibaba, and Tencent, leading to a highly centralized monopoly.

Currently, most content creators rely on these large companies or social media platforms for traffic monetization, but several issues persist, such as:

- **Platform Monopoly and High Commission Rates**: A significant amount of data and users are concentrated on well-known platforms, forcing many creators to use these platforms, which charge high commission fees due to their popularity.
- **High Monetization Threshold & Intense Competition**: Individual creators and small websites struggle to earn substantial income from traffic monetization, as only high-traffic creators receive significant payouts.
- **Fierce Competition for Traffic on Emerging Platforms**: Advertisers prefer to invest in top-tier traffic, making it difficult for smaller creators to secure partnerships.
- **Long Settlement Cycles**: For example, Google Ad-Sense requires a payment cycle of over 30 days, affecting the cash flow of small traffic owners.

1.3.2. **Traffic Fraud and Fake Traffic**

- **Bot Traffic**: Advertisers pay for traffic, but it may actually come from bots or scripts, leading to low conversion rates for ad campaigns.
- **Click Fraud**: Malicious competitors or cheaters use automated programs to click on ads, causing significant losses for advertisers.

1.3.3. **Lack of Transparency & Unfair Settlements**

- Advertisers cannot track the true source of traffic and rely on platform-provided data, which may be falsified or misleading.

1.3.4. **Privacy Issues & Misuse of User Data**

- Centralized platforms collect vast amounts of user data and use AI algorithms for targeted advertising, severely infringing on user privacy.
- Users have no control over their data, and personal information is sold to third-party advertising companies.

1.3.5. **Undervaluation of Traffic**

- The current traffic market is primarily based on CPC (Cost Per Click) and CPM (Cost Per Thousand Impressions), but it fails to effectively measure the true contribution of users.
- User behavior data (such as dwell time and interaction rates) is not effectively monetized, preventing traffic providers from earning fair compensation.

1.4. **What is Traffic Tokenization?**

- The core of traffic tokenization is transforming traffic into freely tradable tokens on the blockchain. Each token represents the traffic property rights of the traffic

provider. Blockchain technology enables this traffic to become a digital asset, allowing for secure, transparent, and seamless global trading.

- Traffic tokenization breaks away from traditional traffic monetization methods, providing better liquidity for content creators and traffic providers. Compared to traditional methods, it significantly lowers the monetization threshold, addressing the issue of unfair earnings for small creators and ordinary individuals, enabling more people to participate.

1.5. **SwiftBlock Coin Blockchain and Technical Architecture**

The SwiftBlock Network is an open-source public blockchain powered by the SwiftBlock Consensus Protocol (SCP), a Proof-of-Agreement (PoA) consensus mechanism. Driven by the SCP, the SwiftBlock Coin network is faster, cheaper, and more energy-efficient than many other blockchains. Transactions are finalized and added to the blockchain once computers, known as "nodes," reach consensus through the SwiftBlock Consensus Protocol. Anyone can set up a SwiftBlock node and participate, but they must provide their identity information in the public record. This allows other nodes to decide whom to include or exclude from their trusted groups. The SCP achieves consensus through a series of voting processes among these mutually trusted nodes. When a sufficient number of nodes in the trusted overlapping group (known as a quorum) agree that a set of transactions and their associated assets and operations are valid, they are permanently added to the blockchain. This process typically takes around 5 seconds.

1.5.1. **Key Features of SCP**

SCP is the first provably secure consensus mechanism with four key attributes:

- **Decentralized Control**: Anyone can participate, and there is no central authority to determine whose approval is needed to reach consensus.
- **Low Latency**: Nodes can reach consensus within the time frame expected for web or payment transactions, typically within a few seconds.
- **Flexible Trust**: Users are free to trust any combination of parties they deem appropriate. For example, a small non-profit organization could play a crucial role in keeping large institutions honest.
- **Asymptotic Security**: Security relies on digital signatures and hash families, whose parameters can be practically adjusted to protect against adversaries with unimaginably vast computational power.

1.5.2. **SCP Consensus Process**

The SCP consensus process is divided into four stages, each ensuring data security and finality:

- **Proposal**: Nodes propose transaction candidates to the network, and all nodes exchange information.
- **Voting**: Voting is conducted through quorum slices to determine the consensus

group.

- **Confirmation**: Once a transaction receives sufficient support, it enters the confirmation state, preventing forks.
- **Application**: The transaction is written to the ledger, finalized, and becomes immutable.

1.5.3. **Main Components of SCP**

- **Nodes**: Servers running the SCP protocol, responsible for transaction validation and consensus.
- **Federated Byzantine Agreement (FBA)**: Consensus is achieved through quorum slices without requiring full network synchronization.
- **Quorum Slices**: Each node independently selects other nodes it trusts, forming a partial trust network.
- **Ledger**: Records all transaction history, similar to a blockchain but with faster updates.
- SwiftBlock **Core**: The main engine that processes the SCP protocol and maintains the blockchain.
- **Horizon API**: A REST API for developers to access the SwiftBlock Network, supporting functions such as transaction queries and account management.

1.5.4. **Characteristics of Different Consensus Mechanisms**

mechanism	decentralized control	low latency	flexible trust	asymptotic security
proof of work	✓			
proof of stake	✓	maybe		maybe
Byzantine agreement		✓	✓	✓
Tendermint	✓	✓		✓
SCP (this work)	✓	✓	✓	✓

There are differences between SwiftBlock Consensus Protocol (SCP) and other consensus mechanisms.The most well-known decentralized consensus mechanism is the Proof of Work (PoW) scheme proposed by Bitcoin. Bitcoin employs a dual-pronged approach to achieve consensus. First, it provides incentives for rational actors to behave well. Second, it settles transactions through the Proof of Work algorithm, which is designed to prevent misbehaving actors who do not possess the majority of the system's computational power.

Bitcoin has fully demonstrated the appeal of decentralized consensus. However, Proof of Work also has its limitations. First, it is resource-intensive—Bitcoin may consume as much electricity as the entire country of Ireland. Second, the expected delay for secure transaction settlement is on the order of minutes or tens of minutes.

Finally, compared to traditional cryptographic protocols, Proof of Work does not provide asymptotic security. When considering irrational attackers or those with external motives to disrupt, even a slight computational advantage can invalidate security assumptions, allowing history to be rewritten in what is known as a "51% attack." Worse, an attacker initially controlling less than 50% of the computational power could manipulate the system by offering disproportionate rewards to those who join them.

An alternative to Proof of Work is Proof of Stake (PoS), where consensus depends on parties who have provided collateral. Like Proof of Work, rewards encourage rational participants to comply with the protocol, while the design also penalizes bad behavior. Proof of Stake opens up the possibility of so-called "nothing-at-stake" attacks, where parties who previously provided collateral but later cashed it out and spent it can return to a point where they still held the collateral and rewrite history. To mitigate such attacks, systems effectively combine Proof of Stake with Proof of Work—either scaling the required work proportionally to the collateral or delaying the return of collateral long enough for other consensus mechanisms to establish irreversible checkpoints.

Another method for achieving consensus is Byzantine Agreement, with the most famous variant being PBFT (Practical Byzantine Fault Tolerance). Byzantine Agreement ensures consensus despite arbitrary behavior (including irrationality) by some participants. This approach has two attractive properties. First, consensus can be achieved quickly and efficiently. Second, trust is completely decoupled from resource ownership, enabling small non-profit organizations to help keep more powerful entities (such as banks or CAs) honest. However, complicating matters is the requirement that all parties must agree on the exact list of participants. Additionally, measures must be in place to prevent attackers from joining multiple times and exceeding the system's fault tolerance, known as Sybil attacks. BFT-CUP accommodates unknown participants but still assumes the existence of a Sybil-resistant centralized admission control mechanism. Generally, membership in Byzantine Agreement systems is determined by a central authority or closed negotiations.

Tendermint takes a different approach by basing membership on Proof of Stake. However, this again ties trust to resource ownership.

SCP is the first Byzantine Agreement protocol that gives each participant maximum freedom in choosing which combinations of other participants to trust.

1.5.5. **Federated Byzantine Agreement (FBA) Model**

The Federated Byzantine Agreement (FBA) model, like non-federated Byzantine Agreement, addresses the problem of updating replicated states (e.g., transaction ledgers or certificate trees). By agreeing on which updates to apply, nodes can avoid

contradictory, irreconcilable states. Each update is identified by a unique slot, from which dependencies between updates can be inferred. For example, a slot might be a sequentially numbered position in a log that is applied in order.

An FBA system runs a consensus protocol to ensure that nodes agree on the contents of slots. When a node *v* has safely applied updates in all slots that *i* depends on, it can safely apply update *x* in slot *i*. Furthermore, it believes that all properly functioning nodes will eventually agree on (X) for slot *i*. At this point, we say that (ν) has externalized *x* for slot *i*. Externalized values may trigger irreversible reactions from the outside world, so nodes cannot change their minds about them later.

A challenge for FBA is that malicious parties can join multiple times, outnumbering honest nodes. Therefore, traditional majority-based quorum systems do not work. Instead, FBA determines quorums in a decentralized manner, with each node selecting what we call a quorum slice. The next subsection defines quorums based on slices. The following subsections provide examples and discussions. Finally, we define the key properties of safety and liveness that a consensus protocol should aim to achieve.

1.5.5.1. **Quorum Slices**

In a consensus protocol, nodes exchange messages about slot declarations. We assume that such declarations cannot be forged, which is guaranteed if nodes are named by public keys and they digitally sign their messages. When a node hears enough nodes assert a declaration, it assumes that no properly functioning node will contradict the declaration. We refer to this "enough nodes" as a quorum slice, or more concisely, just a slice. To allow progress in the presence of node failures, a node can have multiple slices, any one of which is sufficient to convince it to accept a declaration. Thus, at a high level, an FBA system consists of a loose federation of nodes, each of which selects one or more slices.

More precisely: A Federated Byzantine Agreement System (FBAS) is a pair <V, Q>, consisting of a set of nodes V and a quorum function Q: $\nu \rightarrow 2^{2V} \setminus \{\varnothing\}$, which assigns one or more quorum slices to each node. A node belongs to all of its own quorum slices, i.e., $\forall \nu \in V$, $\forall q \in Q(\nu)$, $\nu \in q$. (Note: 2^X denotes the power set of X.)

Definition (Quorum): A set of nodes $U \subseteq V$ in an FBAS <V, Q> is a quorum if and only if $U \neq \varnothing$ and *U* contains a slice for each member, i.e., $\forall \nu \in U$, $\exists q \in Q(\nu)$ such that $q \subseteq U$.

A quorum is a set of nodes sufficient to reach agreement. A quorum slice is a subset of a quorum that convinces a specific node to believe in the agreement.

Quorum slices may be smaller than quorums. Consider the four-node system in Figure 2, where each node has one slice, and arrows point to the other members of the slice. The slice $\{\nu_1, \nu_2, \nu_3\}$ is sufficient for ν_1 to believe a declaration. However, the slices of ν_2 and ν_3 include ν_4, meaning neither ν_2 nor ν_3 can assert a declaration without ν_4's agreement. Therefore, consensus cannot be reached without ν_4's participation, and the only quorum including ν_1 is the set of all nodes $\{\nu_1, \nu_2, \nu_3, \nu_4\}$.

Traditional non-federated Byzantine Agreement requires all nodes to accept the same slices, i.e., $\forall \nu 1, \nu 2, Q(\nu_1) = Q(\nu_2)$. Since every member accepts every slice, traditional systems fail to distinguish between slices and quorums. The drawback is that membership and quorums must be determined in advance in some way, precluding open membership and decentralized control. Traditional systems (e.g., PBFT) typically consist of $3f + 1$ nodes, where any $2f + 1$ nodes form a quorum. Here, f is the maximum number of Byzantine faulty nodes (i.e., nodes behaving arbitrarily) that the system can tolerate.

The key innovation of FBA is enabling each node ν to select its own independent set of quorum slices $Q(\nu)$. Thus, system-wide quorums emerge from the independent decisions of each node. Nodes can choose slices based on arbitrary criteria such as reputation, financial arrangements, etc. In some cases, consensus can still be achieved even if no single node has a complete understanding of all nodes in the system.

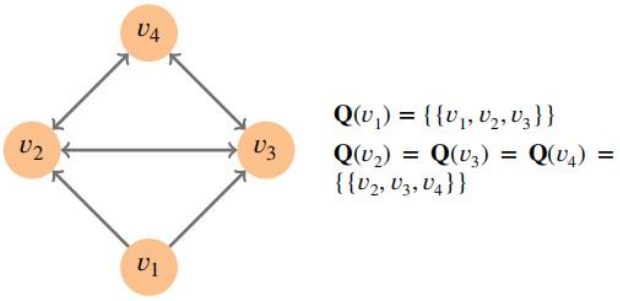

Figure 2: Without ν_4, ν_1's Quorum Slice Would Not Be Considered a Quorum

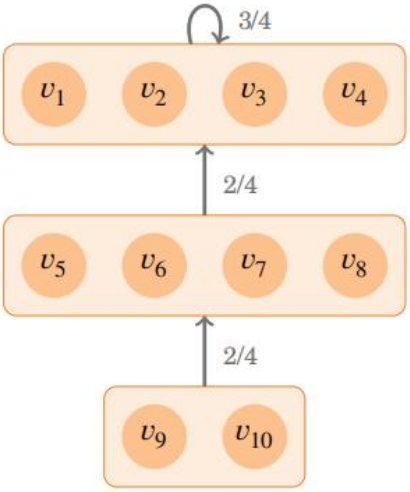

Figure 3: Example of a Hierarchical Quorum Structure

Top Layer: Includes itself, with slices consisting of 3 slices: {v_1, v_2, v_3, v_4}
Middle Layer: Slices consist of itself + any two top-layer nodes
Leaf Layer: Slices consist of itself + any two middle-layer nodes

1.5.5.2. Examples and Discussion

Figure 3 illustrates an example of a hierarchical system where different nodes have different sets of slices, a feature only achievable with FBA. The top layer consists of v_1, ..., v_4, structured similarly to a PBFT system with $f = 1$, meaning it can tolerate one Byzantine fault as long as the other three nodes are reachable and behave correctly. Nodes v_5, ..., v_8 form the middle layer, which does not depend on each other but relies on the top layer. Only two top-layer nodes are needed to form a slice for a middle-layer node. (The top layer assumes at most one Byzantine fault, so unless the entire system fails, two top-layer nodes cannot fail simultaneously.) Nodes v_9 and v_{10} are at the leaf layer, where a slice consists of any two middle-layer nodes. Note that v_9 and v_{10} can choose non-overlapping slices, such as {v_5, v_6} and {v_7, v_8}; nonetheless, both will indirectly depend on the top layer.

In practice, the top layer might consist of 4 to 12 well-known and trusted financial institutions. As the top layer grows, opinions about its membership may not fully align, but most relevant parties will have significant overlap in their understanding of the top layer. Additionally, multiple middle layers could be envisioned, such as one per country or geographic region.

This tiered structure resembles inter-domain network routing. The Internet today is held together by individual peering and transit relationships between pairs of net- works. No central authority dictates or arbitrates these arrangements. Yet these pair- wise relationships have sufficed to create a notion

of de facto tier one ISPs Though Internet reachability does suffer from firewalls, transitive reachability is nearly complete—e.g., a firewall might block The New York Times, but if it allows Google, and Google can reach The New York Times, then The New York Times is transitively reachable. Transitive reachability may be of limited utility for web sites, but it is crucial for consensus; the equivalent example would be Google accepting statements only if The New York Times does.

If we analogize quorum slices to network reachability and quorums to transitive reachability, the internet's near-complete transitive reachability suggests that we can similarly ensure global consensus through FBA. In many ways, solving consensus is simpler than inter-domain routing. While transit consumes resources and incurs costs, slice inclusion only requires checking digital signatures. Thus, FBA nodes can err on the side of inclusion, constructing conservative slices with higher interdependence and redundancy than typically seen in peering and transit arrangements.

Another example impossible in centralized consensus is a cyclic dependency structure, as shown in the figure. Such cycles are unlikely to arise intentionally, but when individual nodes choose their own slices, the system as a whole may end up with dependency cycles. More importantly, compared to traditional Byzantine Agreement, FBA protocols must handle diverse group structures.

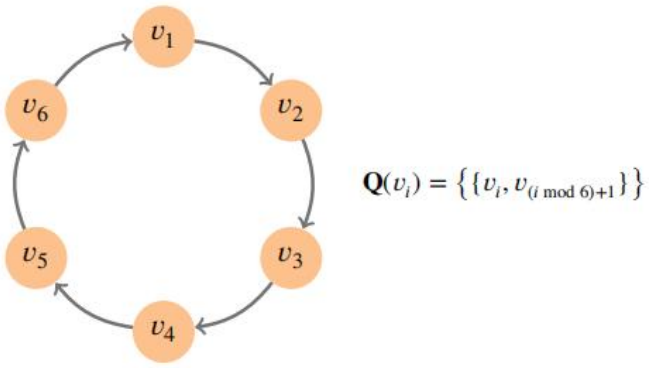

Figure 4: Example of a Cyclic Quorum Structure

1.5.5.3. Safety and Liveness

We categorize nodes into well-behaved and misbehaved nodes. Well-behaved nodes select reasonable quorum slices and follow the protocol, including eventually responding to all requests. Misbehaved nodes, on the other hand, do not. Misbehaved nodes may experience Byzantine faults, meaning their behavior is arbitrary. For example, a misbehaved node might be compromised, its owner might maliciously modify its software, or it might crash.

The goal of Byzantine Agreement is to ensure that well-behaved nodes externalize consistent values even in the presence of misbehaved nodes. This goal encompasses two aspects. First, we want to prevent divergence among nodes and ensure that the same value is externalized for the same slot. Second, we want to ensure that nodes can effectively externalize values without getting stuck in a deadlock where consensus cannot be reached. To this end, we introduce the following two terms:

Definition (Safety): A set of nodes in an FBAS is considered safe if no two nodes externalize different values for the same slot.

Definition (Liveness): A node in an FBAS is considered to have liveness if it can externalize new values without the participation of any faulty (including misbehaved) nodes.

Nodes that are both safe and live are called correct nodes. Misbehaved nodes are faulty. All misbehaved nodes are faulty, but well-behaved nodes can also become faulty if they wait indefinitely for messages from misbehaved nodes or, worse, if their state is affected by incorrect messages from misbehaved nodes.

Figure 5 illustrates the types of node faults that can occur. On the left are Byzantine faults, which are misbehaved nodes. On the right are two types of well-behaved but faulty nodes. Nodes lacking liveness are called blocking nodes, while nodes lacking safety are called divergent nodes. Attacks violating safety are strictly more severe than those violating only liveness, so we consider divergent nodes a subset of blocking nodes. Our definition of liveness is relatively lenient, as it indicates that nodes can externalize new values, not that they necessarily will.

Thus, it acknowledges a state of perpetual preemption, where consensus is always possible, but the network continuously hinders the process by delaying or reordering critical messages in undesirable ways. In purely asynchronous, deterministic systems, perpetual preemption is unavoidable in the presence of node faults. Fortunately, preemption is temporary. It does not imply node failure, as the system can recover at any time. Protocols can mitigate this issue by introducing randomness or making realistic assumptions about message delays. Delay assumptions are more practical when one wishes to bound execution time or avoid the need for a trusted dealer typically required by more efficient randomized algorithms. Of course, only termination—not safety—should depend on message timing.

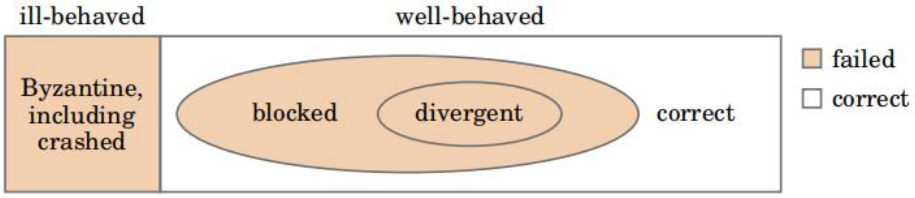

Figure 5: Venn Diagram of Node Faults

1.5.5.4. Optimal Resilience

The safety and liveness of nodes depend on several factors: the quorum slices they choose, which nodes are misbehaving, and the specific consensus protocol and network behavior. As is common in asynchronous systems, we assume that the network will eventually deliver messages between well-behaved nodes, but messages may otherwise experience arbitrary delays or reordering.

This section explores the following question: Given a specific V,Q and a particular subset of misbehaving nodes in V, what is the best safety and liveness that any Federated Byzantine Agreement (FBA) protocol can guarantee, regardless of the network's state? We first discuss quorum intersection, emphasizing that the lack of this property will lead to the inability to guarantee safety. Next, we introduce the concept of dispensable sets, which are sets of faulty nodes that, despite their presence, still allow safety and liveness to be preserved.

1.5.5.5. **Quorum Intersection**

A protocol can only ensure consistency if the quorum slices represented by the function Q satisfy what we call the quorum intersection property.

Definition (Quorum Intersection): An FBAS has quorum intersection if and only if any two quorums share at least one node—that is, for all quorums U_1 and U_2,$U_1 \cap U_2 \neq \varnothing$.

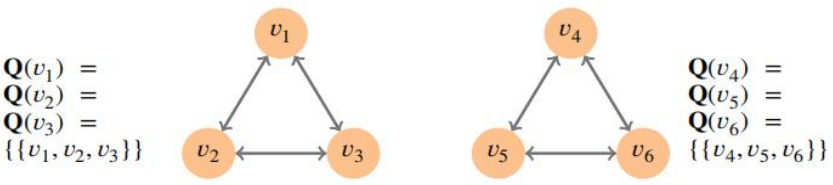

Fig. 6. FBAS lacking quorum intersection

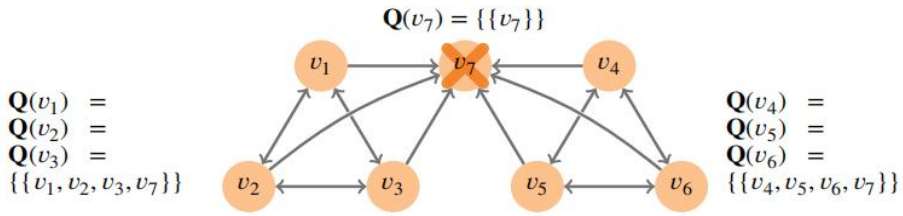

Fig. 7. Ill-behaved node v_7
can undermine quorum intersection.

Figure 6 illustrates a system lacking quorum intersection, where Q allows two disjoint quorums $\{v_1,v_2,v_3\}$ and $\{v_4,v_5,v_6\}$.These disjoint quorums can independently agree on conflicting statements, thereby undermining the consistency of the entire system. When multiple quorums exist, quorum intersection fails if any two quorums are disjoint. For example, the set of all nodes $\{v_1,\ldots,v_6\}$intersects with the other two quorums, but the system still lacks quorum intersection because the other two quorums do not intersect with each other.

Without quorum intersection, no protocol can ensure safety, as this configuration can be viewed as two independent FBAS systems that do not exchange any messages. However, even with quorum intersection, safety may still be compromised in the presence of misbehaving nodes. See Figure 6 (showing two disjoint quorums) and Figure 7 (showing two quorums intersecting at a single nodev7 ,where v_7 is misbehaving). If v_7 makes inconsistent statements to the left and right quorums, the effect is equivalent to having disjoint quorums.

In fact, since misbehaving nodes contribute nothing to safety, no protocol can ensure safety if the well-behaved nodes themselves do not enjoy quorum intersection. After all, in the worst-case scenario for safety, misbehaving nodes can always make any possible (contradictory) statements to satisfy quorums. Because misbehaving nodes are duplicitous, two quorums that overlap only at misbehaving nodes can again operate like two separate FBAS systems. In short, an FBAS <V,Q> can tolerate Byzantine faults by a set of nodes $B \subseteq V$ iff V,Q enjoys quorum intersection after removing the nodes in *B* from V and all slices in Q. More formally:

Definition (Delete): If <V,Q> is an FBAS and $B \subseteq V$ is a set of nodes, then deleting B from <V,Q> denoted $<V,Q>^B$, means computing the modified FBAS $<V \setminus B, Q^B>$, where $Q^B(v)=\{q \setminus B \mid q \in Q(v)\}$.

Each node *v* is responsible for ensuring that Q(*v*) does not violate quorum intersection. One way to achieve this is by choosing conservative slices, thereby forming larger quorums. Of course, a malicious *v* might intentionally select Q(*v*)

to violate quorum intersection. But a malicious v could also falsely report the value of Q(v) or ignore Q(v) to make arbitrary assertions. In short, when v is misbehaving, the value of Q(v) becomes meaningless. This is why the necessary property for safety—quorum intersection among well-behaved nodes after removing misbehaving nodes—is unaffected by the slices of misbehaving nodes.

Suppose Figure 6 evolves from a three-node FBAS $\{\nu_1, \nu_2, \nu_3\}$ with quorum intersection to a six-node FBAS without quorum intersection. When ν_4, ν_5, ν_6 join, they maliciously choose slices that violate quorum intersection, and no protocol can ensure the safety of V. Fortunately, deleting the misbehaving nodes to produce $<V,Q>^{\{\nu4, \nu5, \nu6\}}$ restores quorum intersection, meaning that at least $\{\nu_1, \nu_2, \nu_3\}$ can enjoy safety. Note that deletion is conceptual, intended to describe optimal safety. The protocol should ensure the safety of ν_1, ν_2, ν_3 without requiring them to know about the misbehavior of ν_4, ν_5, ν_6.

1.5.5.5. Dispensable Sets

We capture the fault tolerance of node slice selection through the concept of dispensable sets, or DSets. Informally, a DSet ensures the safety and liveness of nodes outside the DSet, regardless of the behavior of nodes inside the DSet. In other words, in an optimally resilient FBAS, if a single DSet contains every misbehaving node, it must also contain every failed node, and conversely, all nodes outside the DSet are correct.

For example, in a centralized PBFT system with $3f+1$ nodes and a quorum size of $2f+1$, any set of f or fewer nodes constitutes a DSet. Since PBFT can actually tolerate f Byzantine faults, its resilience is optimal.

In less common cases, $\{v_1\}$ is a DSet because the failure of a single top-layer node does not affect the rest of the system. $\{v_9\}$ is also a DSet because no other node depends on the correctness of v_9. $\{v_6,\ldots,v_{10}\}$ is a DSet because neither v_5 nor the top-layer nodes depend on any of these five nodes. However, $\{v_5,v_6\}$ is not a DSet because it is a slice for v_9 and v_{10}, and if fully malicious, it could deceive v_9 and v_{10} into making assertions that are inconsistent with each other or with the rest of the system.

To prevent a misbehaving DSet from affecting the correctness of other nodes, two properties must be satisfied. For safety, deleting the DSet must not break quorum intersection. For liveness, the DSet must not prevent other nodes from functioning properly. This leads to the following definition:

Definition (DSet): Let $<V,Q>$ be an FBAS, and $B \subseteq V$ be a set of nodes. We call B a dispensable set or DSet if and only if:

1. (Quorum Intersection Despite B):$<V,Q>^{B}$ enjoys quorum intersection, and
2. (Quorum Availability Despite B): V∖ B is a quorum in <V,Q> or B=V.

Quorum availability despite *B* prevents nodes in *B* from refusing to respond to requests and blocking the progress of other nodes. Quorum intersection despite *B* prevents the opposite scenario—nodes in *B* making contradictory assertions, allowing other nodes to externalize inconsistent values for the same slot. Nodes must balance these two threats in their slice selection. All else being equal, larger slices lead to stronger quorums and greater overlap, meaning that fewer sets of faulty nodes *B* will break quorum intersection upon deletion. On the other hand, larger slices are more likely to include faulty nodes, jeopardizing quorum availability.

The smallest DSet containing all misbehaving nodes may also include well-behaved nodes, reflecting the fact that a sufficiently large set of misbehaving nodes can cause well-behaved nodes to fail. For example, the smallest DSet containing v_5 and v_6 is $\{v_5,v_6,v_9,v_{10}\}$. The set of all nodes V is always a DSet, as the FBAS $<V,Q>$ vacuously enjoys quorum intersection despite V and, in the special case, quorum availability despite V. The special case is motivated by the fact that, given enough misbehaving nodes, V might be the smallest DSet containing all misbehaving nodes, indicating that no protocol can guarantee a better outcome than complete system failure in such a scenario.

The DSets in an FBAS are determined a priori by the quorum function *Q*. Whether nodes are well-behaved or misbehaving depends on runtime behavior, such as a machine being compromised. The DSets of interest are those that contain all misbehaving nodes, as they help us distinguish nodes that should be guaranteed correctness from those that cannot. To this end, we introduce the following terms:

Definition (Intact): A node v in an FBAS is intact if and only if there exists a DSet *B* containing all misbehaving nodes, and $v \notin B$.

Definition (Contaminated): A node v in an FBAS is considered contaminated if and only if it is not intact.

A contaminated node v is surrounded by enough faulty nodes that, even if v itself is well-behaved, its progress is hindered or its state is corrupted. No FBAS can guarantee the correctness of contaminated nodes. However, an optimal FBAS can ensure that every intact node remains correct. The following theorem facilitates analysis by showing that the set of contaminated nodes is always a DSet in an FBAS with quorum intersection.

well-behaved / ill-behaved	Local property of nodes, independent of other nodes (except for the validity of slice selection).
intact / befouled	Property of nodes given their quorum slices and a particular set of ill-behaved nodes. Befouled nodes are ill-behaved or depend, possibly indirectly, on too many ill-behaved nodes.
correct / failed	Property of nodes given their quorum slices, a concrete protocol, and actual network behavior. The goal of a consensus protocol is to guarantee correctness for all intact nodes.

Figure 8: Key Properties of FBAS Nodes

Theorem 1: Let U be a quorum in the FBAS <V,Q>, let B⊆V be a set of nodes, and let U′=U∖ B. If U′=∅ , then U′is a quorum in $\langle V,Q\rangle^{B}$.

Proof: Since *U* is a quorum, for every node $v \in U$, there exists a $q \in Q(v)$ such that q⊆U. Given that U′⊆U, for every $v \in U'$,there also exists a $q \in Q(v)$ such that q∖ B⊆U′. Rewriting this using the deletion notation, for every q⊆U′ , there exists a ∃q∈QB(*v*) such that q⊆U′. Since U′⊆V∖ B, this shows that U′is a quorum in $\langle V,Q\rangle^{B}$.

Theorem 2: If B_1 and B_2 are DSets in the FBAS <V,Q> with quorum intersection, then $B=B_1 \cap B_2$ is also a DSet.

Proof: Let $U_1=V \setminus B_1$ and $U_2=V \setminus B_2$. If $U_1=\emptyset$, then $B_1=V$ and $B=B_2$ (a DSet), completing the proof. Similarly, if $U_2=\emptyset$, then $B=B_1$, also completing the proof. Otherwise, note that by quorum availability, despite the existence of Sets B_1 and B_2, U_1 and U_2 remain quorums in <V,Q>. By definition, the union of two quorums is also a quorum. Therefore, $V \setminus B=U_1 \cup U_2$ is a quorum, and quorum availability is preserved despite *B*.

We now prove quorum intersection despite *B*. Let U_a and U_b be any two quorums in $\langle V,Q\rangle^{B}$. Let $U=U_1 \cap U_2=U_2 \setminus B_1$. By quorum intersection in <V,Q>, $U=U_1 \cap U_2 \neq \emptyset$. However, by Theorem 1,$U=U_2 \setminus B_1$, so *U* must be a quorum in $\langle V,Q\rangle_{B1}$.

Now consider that $U_a \setminus B_1$ and $U_a \setminus B_2$ cannot both be empty; otherwise, $U_a \setminus B=U_a$ would be empty. Therefore, either $U_a \setminus B_1$ is a quorum in $\langle V,Q\rangle_{B1}$, or $U_a \setminus B_2$ is a quorum in $\langle V,Q\rangle_{B2}$, or both. In the first case, note that if $U_a \setminus B_1$ is a quorum in $\langle V,Q\rangle_{B1}$, then by quorum intersection in $\langle V,Q\rangle_{B1}$, $(U_a \setminus B_1) \cap U \neq \emptyset$. Since $(U_a \setminus B_1) \cap U=(U_a \setminus B_1) \setminus B2$, it follows that $U_a \setminus B_2 \neq \emptyset$, making $U_a \setminus B_2$ a quorum in $\langle V,Q\rangle_{B2}$. By a similar argument, $U_b \setminus B_2$ must also be a quorum in $\langle V,Q\rangle_{B2}$. However, despite the existence of B_2, quorum intersection ensures that $(U_a \setminus B_2) \cap (U_b \setminus B_2) \neq \emptyset$, which is only possible if $U_a \cap U_b \neq \emptyset$.

Theorem 3: In an FBAS with quorum intersection, the set of compromised nodes is a DSet.

Proof: Let $B_{\min}$ be the intersection of all DSets containing all misbehaving nodes. By the definition of integrity, a node v is intact if and only if $v \notin B_{\min}$. Therefore, $B_{\min}$ is precisely the set of compromised nodes. Since DSets are closed under intersection, $B_{\min}$ is also a DSet.

1.6. Core Advantages of SwiftBlock Coin

1.6.1. SCP Advantages

SwiftBlock Coin provides a transparent, secure, user-friendly, and efficient transaction method through the SCP SwiftBlock Consensus Protocol, offering a highly flexible decentralized ecosystem.

- **Transparency**: All transaction records are publicly verifiable, ensuring users can access ledger information at any time.
- **Efficiency**: By optimizing block confirmation time and reducing network latency, SCP achieves faster transaction processing.
- **Security**: A distributed node collaboration mechanism reduces the risk of single points of failure and malicious attacks.
- **User-Friendliness**: Lower transaction costs enable individuals and businesses to participate in the blockchain ecosystem at a lower cost.
- **Decentralized Ecosystem**: Allows users to freely participate in governance, ensuring fairness and sustainability of the ecosystem.

1.6.2. SCP Mechanism Advantages

The (SCP) SwiftBlock Consensus Protocol is an improved consensus mechanism that combines Byzantine Fault Tolerance (PBFT) and Proof of Stake (PoS), known as Proof of Agreement (PoA), aiming to achieve higher security and transaction efficiency.

- **Strong Fault Tolerance**: With BFT integration, the network remains stable even if some nodes fail or act maliciously.
- **Low Energy Consumption**: Compared to Proof of Work (PoW), SCP uses stake-accelerated verification, eliminating the need for extensive computational resources and reducing network energy consumption.
- **High Throughput**: The optimized consensus algorithm reduces unnecessary transaction confirmation steps, significantly increasing TPS (transactions per second).

- **Decentralization**: Removes control by traditional centralized institutions, ensuring network fairness and autonomy.

1.6.3. SwiftBlock Coin Ecosystem Advantages

Tokenizing traffic allows all traffic to become tradable, with SwiftBlock Coin serving as the primary medium for free market transactions.

- **SwiftFi Ecosystem**: SwiftBlock Coin supports smart contract deployment, enabling users to participate in decentralized lending, liquidity mining, stablecoin trading, and other financial activities.
- **NFT Marketplace**: Leveraging SwiftBlock Coin's smart contract mechanism, users can create, trade, and auction NFT assets, facilitating decentralized transactions of digital art, in-game items, and other digital assets.
- **Cross-Chain Interoperability**: SwiftBlock Coin uses cross-chain bridge technology to enable asset swaps with mainstream public chains (e.g., Ethereum, BSC, Polkadot), enhancing liquidity.

1.6.4. Use Cases

SwiftBlock Coin is a revolutionary cryptocurrency focused on tokenizing traffic, offering high flexibility and a wide range of applications.

- **Traffic Tokenization**: Users can easily purchase traffic using SwiftBlock Coin, establishing it as the primary digital currency for traffic.
- **Traditional Business Traffic Trading**: Businesses can buy traffic using SwiftBlock Coin, boosting transaction volumes without relying on third-party intermediaries (e.g., ad agencies, payment channels), reducing costs, and enabling global traffic and borderless payments.
- **Financial Activities**: Through the SwiftFi ecosystem, SwiftBlock Coin enables decentralized financial activities like lending, smart contract investments, and staking, lowering barriers and increasing efficiency.
- **NFT Marketplace**: Users can create, trade, and auction NFT assets using SwiftBlock Coin's smart contract mechanism.
- **Decentralized Traffic Exchange (DAE)**: A blockchain-based, intermediary-free ad exchange market using SwiftBlock Coin as its core digital asset, enhancing transparency, security, and liquidity while reducing ad costs.

1.6.5. Asset Security

The (SCP) SwiftBlock Consensus Protocol integrates multiple algorithms and mechanisms to achieve higher security.

- **Decentralized Consensus**: Blockchain technology ensures digital asset security through decentralized consensus and rapid finality.
- **Flexible Trust Model**: SCP's flexible trust mechanism allows nodes to choose trusted partners and validators based on their needs, enhancing overall network security.
- **No Dependency on Super Nodes**: Even with a few malicious nodes, consensus security is maintained, ensuring network operation despite DDoS attacks, hacks, or node failures.
- **Low Latency**: Transactions are confirmed within seconds without relying on miners or computational power, avoiding security risks from delays.
- **No Smart Contract Vulnerabilities**: SwiftBlock Coin does not support Turing-complete smart contracts, eliminating common attacks like reentrancy or integer overflow.

1.6.6. SFB Reward Mechanism

SwiftBlock Coin builds a decentralized, transparent, and fair incentive ecosystem, rewarding users for ad engagement, content creation, and traffic trading.

- **Users**: Earn SwiftBlock Coin by watching ads, interacting, and sharing.
- **Advertisers**: Receive data analysis rewards for ad bidding and placement.
- **Traffic Providers**: Earn SwiftBlock Coin for precise recommendations.
- **Data Contributors**: Profit from authorized data sharing.
- **Community Rewards**: Accelerated rewards for community support and growth.

1.6.7. SwiftBlock Coin Distribution

Category	Amount (Coins)	Percentage	Purpose
Internal Network Phase	300,000,000	10%	Global distribution, halving rewards to incentivize early users and growth.
Mainnet Phase	1,950,000,000	65%	Global distribution to drive growth and ecosystem development.
Liquidity & Market Operations	300,000,000	10%	Support market circulation.
Research Team	150,000,000	5%	Fund technical research.
Development Team	150,000,000	5%	Cover development costs.
Incentive Programs	150,000,000	5%	Reward activities and community development.
Total Supply	**3,000,000,000**	**100%**	

1.6.7.1. Total Supply

SwiftBlock Coin has a fixed total supply of 3,000,000,000 coins, ensuring scarcity and long-term value. The distribution is transparent and fair, supporting sustainable growth. The burn mechanism reduces circulating supply, maintaining value stability.

Internal Network Phase

During the internal network phase, SwiftBlock Coin adopts a globally distributed release model to ensure fair and just token distribution. The goal of this phase is to encourage user participation through early incentive mechanisms, driving initial project growth and ecosystem development. By gradually reducing rewards, the project aims to maintain the enthusiasm of early users while preventing excessive inflation. These tokens can be used to reward active users, support community governance, and provide necessary support for the ecosystem. Additionally, a portion of the funds will be allocated to marketing efforts to enhance the visibility of SwiftBlock Coin, attracting more users to understand and participate in the project, thereby laying the foundation for the subsequent mainnet development. To promote a healthy token circulation, SwiftBlock Coin has established a burning mechanism where a portion of tokens is burned during each node registration or on-chain transaction, thereby reducing the total circulating supply and maintaining the token's scarcity and value stability over the long term.

The consensus algorithm of SwiftBlock Coin relies on trustworthy nodes, making it crucial to incentivize pioneers to form personal security circles. This means that before the mainnet launch, the Swift Block Network will offer substantial rewards to high-quality users who help maintain or contribute to community development. Maintaining long-term network incentives is also essential, so the network employs a unique mechanism where the release speed is halved whenever the amount of coins a user receives from node releases doubles their initial investment, and this process repeats until a cap is reached.

Mainnet Phase

The mainnet phase is a critical period for the development of the SwiftBlock Coin ecosystem. The primary purpose of the tokens in this phase is to promote the long-term growth of the entire network, enhance the diversity of decentralized application scenarios, and encourage more users and developers to participate through incentive mechanisms. SwiftBlock Coin ensures fair distribution through a globally distributed release model while optimizing incentive strategies to strengthen users' long-term holding intentions. As the mainnet stabilizes, SwiftBlock Coin will be widely used in various decentralized finance (SwiftFi) applications, payment systems, smart contract platforms, and

more. Additionally, these tokens will support the developer ecosystem by rewarding contributors and funding innovative projects, further improving the Swift Block Network ecosystem. SwiftBlock Coin also implements a "transaction burn" mechanism, where a portion of tokens is burned with each on-chain transfer to reduce circulating supply and ensure steady market value growth.

Liquidity and Market Operations

Liquidity is crucial for the stability and sustainable growth of SwiftBlock Coin in the market. Therefore, a portion of the tokens is allocated to enhance market liquidity, support trading platform operations, and conduct necessary market promotion activities. Good liquidity helps reduce market volatility, enabling investors and users to trade more smoothly while enhancing SwiftBlock Coin's competitiveness in the global market. In terms of market operations, these funds will be used for brand building, community promotion, and establishing partnerships. For example, by collaborating with mainstream exchanges, SwiftBlock Coin aims to increase trading depth and accessibility, making it more widely adopted. Additionally, to further stabilize the market, SwiftBlock Coin's burning mechanism will play a role in liquidity management, where a portion of tokens is permanently burned during market transactions to optimize supply and demand, enhancing the token's long-term value.

Research Team

The continuous development of the SwiftBlock Coin ecosystem relies on strong technical research support. Therefore, a portion of the tokens is allocated to fund the research team's long-term efforts in driving innovation in blockchain core technologies. The research team's main tasks include optimizing blockchain protocols, studying smart contract security, and developing new consensus mechanisms. Through ongoing technical research, SwiftBlock Coin aims to enhance network security, scalability, and efficiency to meet the demands of future decentralized applications. Additionally, these funds will support academic collaborations with universities and research institutions to explore blockchain technology development and promote industry standards. To optimize SwiftBlock Coin's economic model, the research team will refine the burning mechanism to achieve the best balance between network stability and incentives, ensuring the ecosystem's sustainable development.

Technical Team

The technical team is responsible for the development, maintenance, and optimization of the SwiftBlock Coin network. A portion of the tokens is allocated to pay developer salaries, infrastructure construction, and technical upgrades. As an innovator in decentralized finance (SwiftFi) and blockchain technology, SwiftBlock Coin must maintain high technical standards to ensure network

stability and security. The technical team's core tasks include network architecture optimization, node management, smart contract upgrades, and user experience improvements. Additionally, these funds will support the development of the technical community, attracting more developers to join the SwiftBlock Coin ecosystem and contribute to the project's growth. Notably, SwiftBlock Coin employs a dynamic burning mechanism for node registration, where a certain number of tokens are burned with each new node joining, ensuring fair resource allocation and enhancing the token's long-term scarcity.

Incentive Mechanisms

A portion of the tokens is allocated for market promotion, community activities, and incentive programs to increase user engagement and brand influence. By organizing global marketing campaigns, such as online and offline promotions, social media marketing, and community reward programs, SwiftBlock Coin aims to attract more users and businesses to adopt its network. Additionally, these funds will be used to host hackathons, developer competitions, and other events to encourage developers to build innovative applications on SwiftBlock Coin. The project will also introduce a "burn-to-earn" mechanism in incentive activities, where a portion of marketing expenses will be used to purchase and burn tokens during specific campaigns, controlling market supply and enhancing the token's value growth potential.

Burning Mechanism

To ensure the long-term stability of the SwiftBlock Coin ecosystem and the token's scarcity, SwiftBlock Coin implements an automatic burning mechanism where a certain number of tokens are burned during each on-chain transaction or node registration. The core goal of this mechanism is to control market supply, reduce inflation, and enhance the market value of SwiftBlock Coin, making it a truly sustainable cryptocurrency.

Transaction Burning Mechanism

Every on-chain transfer of SwiftBlock Coin burns a portion of tokens. This means that whenever users transact, pay, exchange, or transfer tokens on the network, the system automatically burns a portion of tokens, permanently reducing the circulating supply. This mechanism not only stabilizes the token's price but also incentivizes users to hold tokens long-term, reducing market sell pressure and ensuring steady value growth.

Node Registration Burning Mechanism

To ensure fairness and decentralization in the SwiftBlock Coin network, a burning mechanism is also triggered during each new node registration. New

nodes must pay a certain amount of SwiftBlock Coin as a registration fee, a portion of which is burned. This mechanism prevents network resource abuse, ensuring that only users genuinely committed to network security and stability can participate in node operations, while further reducing circulating supply and enhancing the token's long-term value.

Long-Term Deflationary Effect

SwiftBlock Coin's burning mechanism is a continuous, dynamic process. As network users and transaction volumes grow, the number of tokens burned will gradually increase, creating a long-term deflationary effect. This mechanism ensures that SwiftBlock Coin avoids inflation issues over time, allowing its value to grow steadily. Additionally, SwiftBlock Coin may introduce periodic buyback and burn strategies to further optimize supply and demand, maintaining the token's market value.

The burning mechanism is a crucial component of SwiftBlock Coin's economic model. Through the dual mechanisms of "transaction burning" and "node registration burning," the token supply is gradually reduced, enhancing its long-term value and market stability. This mechanism not only strengthens SwiftBlock Coin's appeal as a decentralized financial asset but also provides a solid economic foundation for the ecosystem's sustainable development. As the SwiftBlock Coin network expands, the burning mechanism will play an increasingly important role in maintaining token scarcity and stability, making it a truly efficient, secure, and decentralized blockchain ecosystem.

Total Supply

SwiftBlock Coin has a fixed total supply of 3,000,000,000 tokens to ensure scarcity and long-term value. All token allocations follow transparent and fair principles to support the project's sustainable development. Through a rigorous economic model and scientific token release mechanisms, SwiftBlock Coin avoids inflation and price volatility, ensuring stable market value growth. The burning mechanism plays a vital role in the SwiftBlock Coin ecosystem, including burning tokens during on-chain transactions, node registrations, and market operations. This mechanism ensures the long-term stability of SwiftBlock Coin, reduces circulating supply, maintains supply-demand balance, and enhances the token's market value. SwiftBlock Coin is committed to building a secure, efficient, and decentralized blockchain ecosystem, making it a trusted digital asset for global users. In the future, SwiftBlock Coin will continue to optimize its economic model and burning mechanisms to ensure sustainable value growth and promote the widespread adoption of decentralized financial services worldwide.

1.6.8. Future Development

SwiftBlock Coin aims to enhance network security, optimize user experience, expand use cases, and improve market stability. It will upgrade blockchain infrastructure, support complex DApps, and foster global partnerships. The burn mechanism will continue to create deflationary pressure, ensuring long-term value growth.

1.6.9. Conclusion

SwiftBlock Coin is a decentralized cryptocurrency built on blockchain technology, offering an efficient, secure, and transparent digital financial ecosystem. Its SCP SwiftBlock Consensus Protocol ensures high transaction efficiency and network security. Through global distribution and innovative incentives, SwiftBlock Coin promotes ecosystem growth. With a robust economic model and burn mechanism, SwiftBlock Coin aims to become a leading global digital asset, providing secure, efficient, and transparent financial services in a truly open and decentralized blockchain ecosystem.