Winsdet: A Settlement Layer for Scalable Digital Payments

Version 1.0
October 15, 2025

Author: Winsdet Development Team
Website: www.winsdet.com

Abstract

We present Winsdet, a blockchain protocol that addresses scalability through native integration of payment channels into the consensus layer while maintaining deterministic settlement guarantees. The system introduces SPARK (Simple Payments And Rebalancing Kernel), a payment channel mechanism embedded as first-class transaction types within the protocol rather than constructed as external smart contracts. This architectural decision enables consensus-validated state transitions, eliminating entire categories of security vulnerabilities present in retrofitted solutions. Light clients achieve cryptographic verification through receipt proofs without maintaining full blockchain state, enabling genuine mobile deployment. The protocol enforces a mathematically precise monetary supply of one billion units through deterministic halving, with storage sustainability achieved through body pruning and cryptographic state commitments. By separating high-frequency payment activity from on-chain settlement, Winsdet demonstrates that blockchain systems can achieve practical scalability without compromising decentralization or auditability.

---

1. Introduction

1.1 Background

In 2008, Satoshi Nakamoto introduced Bitcoin, demonstrating for the first time that electronic cash could operate as a peer-to-peer network without reliance on trusted financial intermediaries. The fundamental innovation combined digital signatures for ownership control with proof-of-work consensus to solve the double-spending problem in a distributed environment. Bitcoin's unspent transaction output model provided strong guarantees about coin ownership and state transitions, establishing a foundation that subsequent blockchain systems have built upon. The Bitcoin network has successfully processed hundreds of millions of

transactions over fifteen years, demonstrating remarkable resilience and security. This empirical track record validates the core architectural decisions around proof-of-work consensus and the UTXO model. However, practical deployment has revealed fundamental constraints. The Bitcoin network processes approximately seven transactions per second, far below the throughput required for global payment infrastructure. Traditional payment networks handle thousands of transactions per second during normal operation and must scale to handle peak loads during major shopping periods.

1.2 The Scalability Challenge

Attempts to increase blockchain throughput through larger blocks or faster block production have consistently encountered the same fundamental tradeoff. Higher throughput increases resource requirements for full node operation, creating centralization pressure. If only well-resourced entities can afford to validate the chain, the system has merely substituted one form of centralization for another. This observation has led researchers to explore alternative approaches, including payment channels, sidechains, and sharding proposals.

The Lightning Network, introduced by Poon and Dryja, demonstrated that payment channels could enable high-frequency transactions while settling periodically on-chain. This approach preserves the security model of the base layer while dramatically increasing practical throughput. However, Lightning operates as an external protocol layered atop Bitcoin, constrained by the limited scripting capabilities available for implementing complex state machines. Channel operations require intricate multi-step protocols, and security depends on participants maintaining persistent watchtower services to monitor for fraudulent state publications.

1.3 Design Philosophy

Winsdet takes a different approach by integrating payment channels directly into the consensus layer. Rather than constructing channels through script composition, the protocol defines channel lifecycle operations as first-class transaction types with deterministic validation rules. This architectural decision yields several advantages. Consensus nodes directly validate channel state transitions, eliminating ambiguity about valid operations. The protocol can optimize channel operations since they are known primitives rather than general script executions. Security analysis becomes simpler since channel behavior is specified explicitly rather than emerging from script interactions.

The system is designed around several core principles. First, determinism takes precedence over flexibility. Every rule must be mathematically specified without ambiguity. Second, the protocol maintains a minimal surface area, including only mechanisms essential for settlement. Additional functionality belongs at higher layers where it can evolve independently. Third, light client support is treated as a first-class concern rather than an afterthought. A system is only as decentralized as its most resource-constrained participant can meaningfully verify. Fourth, storage requirements must remain bounded or the system will inevitably centralize. Pruning and snapshot mechanisms are protocol-level features, not optional optimizations.

1.4 Contributions

This paper makes several contributions. We present SPARK, a payment channel mechanism with native consensus integration that provides deterministic state transitions and settlement guarantees. We describe a receipt-based verification system that enables light clients to validate transaction inclusion and state commitments without maintaining full blockchain state. We specify a deterministic monetary policy with mathematical precision, eliminating any ambiguity in total supply. We demonstrate how proper separation between settlement and payment layers enables practical scalability while preserving decentralization. Finally, we provide security analysis showing that the system maintains safety and liveness properties under standard honest majority assumptions.

1.5 Acknowledgments

This work builds upon foundational contributions from numerous researchers and practitioners. Satoshi Nakamoto introduced proof-of-work consensus and the UTXO model that underpins our approach. The Lightning Network demonstrated the viability of payment channels for scaling transaction throughput. Academic research on state channels and cryptographic commitments, including work on Merkle trees by Merkle and Sparse Merkle Trees by various researchers, provides the foundation for our state commitment scheme. The BLAKE3 hash function, developed by O'Connor, Aumasson, Neves, and Wilcox-O'Hearn, offers performance improvements over SHA-256. The Ed25519 signature scheme, designed by Bernstein and colleagues, provides efficient and secure digital signatures. The canonical encoding properties of CBOR, standardized in RFC 8949, ensure deterministic serialization. The Linearly Weighted Moving Average difficulty adjustment algorithm, pioneered by Zawy, offers responsive and stable target adjustment. We are grateful to all contributors whose work has made this system possible.

---

2. System Architecture

2.1 Overview

Winsdet employs a layered architecture that separates settlement concerns from payment concerns. The base layer maintains a blockchain secured by proof-of-work consensus, recording state transitions and providing finality. The payment layer consists of SPARK channels that enable high-frequency off-chain transactions while relying on the base layer for dispute resolution and final settlement. Light clients form a third layer, enabling resource-constrained participants to verify the system state through cryptographic proofs without maintaining complete transaction history.

The system maintains three distinct namespaces within its state model. The accounts namespace tracks individual address balances and transaction nonces, following an

account-based approach similar to the UTXO model but with explicit state representation. The channels namespace records SPARK channel state, including participant identities, capacity allocations, current balances, and sequence numbers for dispute resolution. The fee cells namespace supports a sponsored fee mechanism allowing third parties to pay transaction costs, which proves essential for channel operations during periods of fee market volatility.

2.2 State Commitments

Rather than requiring all nodes to maintain complete transaction history, Winsdet commits to current state through Sparse Merkle Trees. Each of the three namespaces maintains its own Merkle tree with 256-bit depth, providing sufficient space to prevent collisions while enabling efficient proof construction. State commitments use domain-separated hashing to prevent cross-tree attacks. Leaf nodes include both the key and value, while internal nodes commit only to child hashes. Empty subtrees are represented by precomputed constant values, enabling sparse storage where only populated nodes require memory.

The block header commits to all three state roots along with the transaction Merkle root. This enables nodes to verify state transitions by applying transactions to a candidate state and confirming the resulting root matches the header commitment. Importantly, this design permits body pruning. After sufficient confirmations, nodes can discard transaction bodies while retaining headers and current state. New nodes can bootstrap from cryptographically verified snapshots rather than replaying entire history, dramatically reducing synchronization time and storage requirements.

2.3 Transaction Types

The protocol defines a limited set of transaction types, each serving a specific purpose. PAY transactions transfer value between accounts, representing simple payments. PAYM transactions support multiple outputs from a single input, enabling exchanges and payment processors to batch withdrawals efficiently. The channel-specific transactions include CH_OPEN for establishing new channels, CH_CLOSE_INIT for initiating unilateral closure, CH_CLOSE_UPDATE for posting higher sequence states during disputes, CH_CLOSE_FINAL for completing settlement, and CH_EXPAND for third-party channel rebalancing.

Each transaction type uses canonical CBOR encoding with strict validation rules. The protocol rejects indefinite lengths, floating point numbers, non-minimal integer encodings, duplicate map keys, and other sources of encoding ambiguity. This ensures that transaction identifiers are deterministic and non-malleable. Transaction IDs are computed by hashing the canonical encoding of the transaction body excluding signatures. Since signatures do not affect the transaction ID, third parties can add or modify signatures without changing transaction identity, which proves essential for fee sponsorship mechanisms.

2.4 Block Structure

Blocks consist of a header and body. The header contains the previous block hash, three state roots for the account, channel, and fee cell namespaces, the transaction Merkle root, a timestamp, a difficulty target encoded compactly, and an eight-byte nonce for proof-of-work. The header structure is intentionally minimal, including only fields necessary for consensus validation. The body contains the coinbase transaction followed by an ordered list of regular transactions.

Block size is limited to 128 kilobytes as a hard consensus rule, with policy guidance suggesting 64 to 100 kilobyte target fills. This conservative approach maintains propagation efficiency while providing sufficient capacity for settlement operations. At ten-second block intervals with 100 kilobyte average size, the system supports approximately one thousand transactions per block. This represents settlement capacity rather than payment capacity. The actual payment throughput depends on the number of active SPARK channels, each of which can process unlimited off-chain transactions.

---

3. SPARK Payment Channels

3.1 Motivation

Payment channels solve a fundamental limitation of blockchain systems. Recording every payment on-chain provides strong security guarantees but creates throughput bottlenecks. Payment channels enable two parties to transact unlimited times with only periodic on-chain settlement. The parties establish a channel by locking funds on-chain, conduct numerous off-chain payments by exchanging signed state updates, and finally settle on-chain to distribute the current balances.

Existing payment channel implementations, notably the Lightning Network, construct channels through complex multi-step protocols using time-locked transactions and penalty mechanisms. While effective, this approach introduces significant complexity in implementation and security analysis. SPARK takes a different approach by making channel operations first-class citizens of the protocol. The consensus layer directly validates channel state transitions, reducing implementation complexity and enabling formal verification of correctness properties.

3.2 Channel Lifecycle

A SPARK channel progresses through several states from establishment to closure. Initially, two parties agree on channel parameters including capacity contributions from each party and a timeout period for dispute resolution. They create a CH_OPEN transaction that locks the agreed capacity in an escrow address controlled by the channel. The transaction includes public keys for both parties, capacity allocations, and the agreed timeout period. Once confirmed on-chain, the channel enters the OPEN state.

In the OPEN state, the parties can perform unlimited off-chain payments by exchanging signed state updates. Each update includes the channel identifier, a strictly increasing sequence number, and the current balance distribution between the two parties. Both parties sign each state update, creating a bilateral commitment. The sequence number provides a total ordering of states, which becomes critical during dispute resolution. Importantly, no state updates touch the blockchain during normal operation. The parties simply maintain the latest signed state locally.

When either party wishes to close the channel, they have two options. In the cooperative case, both parties agree on the final state and co-sign a CH_CLOSE_FINAL transaction. Since both parties explicitly approve the outcome, settlement occurs immediately upon confirmation without requiring a timeout period. This represents the optimal case with minimal on-chain footprint and no delay.

If one party becomes unresponsive or the parties cannot agree on closure terms, either party can initiate unilateral closure by posting a CH_CLOSE_INIT transaction containing what they believe to be the latest state. This transaction includes the channel identifier, sequence number, balance distribution, and signatures from both parties on that particular state. The transaction commits to a specific balance allocation but does not immediately settle. Instead, the channel enters a CLOSING state with a timeout period during which the counter-party can respond.

If the initiating party posted an outdated state, perhaps attempting to claim funds they no longer possess, the counter-party can respond by posting a CH_CLOSE_UPDATE transaction with a higher sequence number. The consensus rules enforce a simple principle: the highest valid sequence number wins. A CH_CLOSE_UPDATE with sequence number n+1 supersedes a CH_CLOSE_INIT with sequence number n. The protocol allows multiple rounds of updates, always selecting the state with the highest sequence number that both parties signed.

After the timeout period elapses with no further updates, the channel can be finalized through a CH_CLOSE_FINAL transaction. This transaction references the channel identifier and triggers distribution of funds according to the highest sequence state posted during the dispute window. The finalization transaction requires only one signature from either party since the dispute period has already established the correct state.

3.3 Security Properties

The SPARK mechanism provides several security guarantees. First, channel capacity is locked cryptographically and cannot be accessed except through valid channel operations signed by appropriate parties. Second, the latest-state-wins rule ensures that the most recent mutually agreed state determines the final outcome. Neither party can profit by posting outdated states since the counter-party can always respond with a higher sequence update. Third, the timeout period provides sufficient opportunity for honest parties to respond to any fraudulent closure attempts.

The security model differs from penalty-based systems where publishing old state results in loss of all funds. In SPARK, publishing old state simply fails to achieve the desired outcome since the counter-party posts a newer state. This eliminates the need for persistent watchtower monitoring in the base protocol, though parties may still choose to run watchtower services for convenience. The key insight is that security derives from the ability to post a higher sequence state, not from the threat of punishment.

The protocol enforces several consensus rules to maintain these properties. Channels must be opened with positive capacity contributions from at least one party. During closure, the system accepts at most one UPDATE transaction per channel per block, preventing rapid-fire state updates that might exploit network propagation delays. A CH_CLOSE_FINAL transaction is invalid if the same block contains a CH_CLOSE_UPDATE with higher sequence number for the same channel, eliminating race conditions during settlement.

3.4 Fee Sponsorship

Payment channels face a unique challenge during closure. The party initiating closure must pay transaction fees to confirm the closure transaction on-chain. If fees spike unexpectedly, the party may lack sufficient funds outside the channel to cover fees, creating a deadlock. The channel holds funds but they cannot be accessed without first closing the channel, which requires paying fees.

Winsdet solves this through inline fee sponsorship. Channel closure transactions include a FeeEnv structure that specifies who pays fees for the operation. The fee payer can be either channel participant or a completely unrelated third party. The fee payer signs their commitment to pay fees independently from channel state signatures. This separation allows payment processors, merchants, or liquidity providers to sponsor closure fees without gaining any control over channel state.

Fee sponsorship proves particularly valuable during high-fee periods or when one party's funds are primarily locked in channels. A merchant running multiple customer channels can sponsor closure fees for all customers, guaranteeing that channels can close regardless of on-chain fee conditions. This removes a significant barrier to channel adoption and prevents fee-pinning attacks where an adversary deliberately inflates fees to prevent channel closure.

3.5 Channel Expansion

Channels are initially funded with fixed capacity contributions from participants. However, payment patterns may cause one party's balance to become depleted over time, limiting the channel's utility. Traditional channel networks address this through routing payments across multiple channels, but this introduces complexity and potential privacy concerns.

SPARK includes a CH_EXPAND operation enabling third-party channel rebalancing. A sponsor can add capacity to an existing channel, with the additional funds credited to either party's balance. The expansion requires signatures from both channel participants and the sponsor,

ensuring all parties consent to the operation. This mechanism allows payment processors to inject liquidity into merchant channels without creating new channels, improving capital efficiency.

Channel expansion also enables interesting business models. A payment processor might offer to maintain customer channel liquidity in exchange for service fees. The processor monitors channel balances and proactively adds capacity before channels become unbalanced, ensuring customers can always send and receive payments. This provides a better user experience than traditional channel management strategies while maintaining the security properties of the base protocol.

---

4. Light Client Verification

4.1 The Verification Problem

Full nodes download every block and transaction, validating all state transitions from genesis. This provides maximum security but imposes significant resource requirements. As of this writing, the Bitcoin blockchain exceeds 500 gigabytes despite aggressive optimizations. Most users cannot or will not maintain this data volume, particularly on mobile devices where storage and bandwidth are constrained.

Simplified Payment Verification, introduced in the Bitcoin whitepaper, enables light clients to verify transaction inclusion by downloading block headers and requesting Merkle proofs from full nodes. However, SPV has fundamental limitations. Light clients cannot verify transaction validity, only that miners included the transaction in a block. If miners collude to include invalid transactions, SPV clients cannot detect the fraud. Additionally, SPV provides no mechanism to verify current account balances or channel states without trusting full nodes to report state accurately.

4.2 Receipt-Based Verification

Winsdet extends SPV concepts through cryptographic receipts that prove both transaction inclusion and state validity. A receipt consists of several components. First, the transaction itself with its deterministic identifier. Second, a Merkle path connecting the transaction to the transaction root in the block header. Third, the block header containing the transaction. Fourth, a chain of subsequent block headers demonstrating proof-of-work accumulation. Fifth, Merkle proofs connecting relevant state elements to the state roots committed in the header.

Light clients use receipts to verify payments without downloading full blocks or maintaining complete state. To verify receipt of a payment, the client checks that the transaction appears in a block header chain with sufficient accumulated proof-of-work. The client then verifies that the transaction updates the recipient's account balance using a state proof connecting the account

record to the committed state root. If both checks pass, the client knows the transaction was both included in the blockchain and resulted in a valid state transition.

This verification model provides substantially stronger security than traditional SPV. Light clients can detect invalid state transitions since any mismatch between the transaction's claimed effects and the committed state roots would cause verification to fail. Miners cannot convince light clients to accept invalid payments without also controlling majority hashrate to produce fraudulent proof-of-work chains, which represents an attack on the base consensus mechanism rather than a flaw in the light client protocol.

4.3 Deployment Scenarios

Receipt-based verification enables several important use cases. Mobile wallets can verify received payments without downloading blocks, using a few kilobytes of receipt data rather than megabytes of block data. This makes genuine mobile deployment practical rather than theoretical. Point-of-sale terminals can verify payment completion instantly by checking receipt proofs against locally cached block headers. Exchanges can verify customer deposits through receipts without running full nodes, reducing operational overhead while maintaining security.

The receipt mechanism also supports interesting privacy properties. Since receipts prove statements about specific transactions and state elements without revealing unrelated information, a party can selectively prove facts to third parties without exposing their complete transaction history. A business can prove to an auditor that specific payments occurred and updated account balances correctly without revealing other transactions in the same blocks.

Light client verification works particularly well with SPARK channels. Once a channel is established on-chain with proper receipts, the parties can conduct unlimited off-chain payments without touching the blockchain. When the channel eventually closes, new receipts prove the final settlement. From a verification perspective, a channel that processes millions of payments generates the same proof burden as two on-chain payments: one for opening and one for closing.

4.4 Snapshot Mechanisms

New nodes face a bootstrapping challenge. They must validate the entire blockchain history to achieve the same security level as long-running nodes. With years of transaction history, this process can take days even with modern hardware. Winsdet addresses this through cryptographic snapshots that enable nodes to begin validating from a recent state rather than from genesis.

A snapshot consists of the current state trees for all three namespaces, the block header at which the snapshot was created, and signatures from a quorum of publisher keys embedded in the genesis configuration. Nodes verify that the snapshot signatures are valid, that the state roots match the header commitments, and that sufficient subsequent blocks confirm the

snapshot height. If these checks pass, the node can begin validating from the snapshot point forward without replaying historical transactions.

Snapshot security depends on the assumption that the publisher quorum remains honest and that the network achieves consensus on the canonical chain. If these assumptions hold, snapshots provide equivalent security to full validation while dramatically reducing synchronization time. New nodes can become fully validating participants within minutes rather than days, lowering barriers to network participation and improving decentralization.

---

5. Consensus and Security

5.1 Proof-of-Work

Winsdet employs Nakamoto consensus through proof-of-work, following the model established by Bitcoin. Miners compete to find block headers whose hash values fall below a dynamically adjusted target. The hash function is BLAKE3, a modern cryptographic hash offering better performance than SHA-256 while maintaining equivalent security properties for collision resistance and preimage resistance.

Block headers are exactly 116 bytes, consisting of the previous block hash (32 bytes), transaction root (32 bytes), three state roots (96 bytes total), timestamp (8 bytes), compact difficulty target (4 bytes), and nonce (8 bytes). The header is hashed as a single byte string without padding or alignment. The resulting hash is interpreted as a 256-bit big-endian integer and compared against the target. Valid blocks satisfy the condition that the hash value is less than or equal to the target.

The difficulty target adjusts every block to maintain an average inter-block interval of ten seconds. This contrasts with Bitcoin's two-week adjustment periods, enabling more responsive adaptation to hashrate changes while introducing new stability requirements. The adjustment mechanism is described in the following section.

5.2 Difficulty Adjustment

Winsdet employs the Linearly Weighted Moving Average algorithm with a 20-block window, commonly abbreviated as LWMA-20. This algorithm, developed by Zawy and refined through deployment in multiple production blockchains, provides superior performance compared to Bitcoin's difficulty adjustment algorithm in scenarios with volatile hashrate.

The algorithm computes difficulty as follows. For each of the previous 20 blocks, calculate the solve time as the difference between consecutive block timestamps. Apply per-step clamping to limit individual solve times to the range from one to six times the target interval. This prevents extreme outliers from skewing the average. Compute a linearly weighted average of the

clamped solve times, where recent blocks receive higher weight than older blocks. Adjust the difficulty target based on the ratio between the weighted average solve time and the target interval.

Additional safeguards prevent manipulation. The overall difficulty adjustment is clamped to ±25% per block, preventing drastic swings that might enable temporal attacks. Block timestamps must exceed the median of the previous 11 block timestamps, preventing miners from manipulating timestamps to influence difficulty calculations. Timestamps cannot exceed the median timestamp plus two hours, preventing unrealistic future timestamps.

These mechanisms ensure that difficulty tracks hashrate relatively accurately even during periods of high volatility. Empirical testing across multiple blockchains has demonstrated LWMA-20's stability and resistance to manipulation. The algorithm responds quickly enough to prevent extended periods of slow or fast blocks while avoiding oscillations that might enable difficulty manipulation attacks.

5.3 Fork Choice

The canonical blockchain is the valid chain with the greatest cumulative proof-of-work. When nodes receive competing blocks at the same height, they provisionally accept the first received but monitor for alternatives. If a longer chain emerges, nodes reorganize to follow the heavier chain. This follows standard Nakamoto consensus without additional finality mechanisms or checkpoints.

Reorganizations deeper than six blocks are unusual under normal operation, occurring primarily during network partitions or sustained attacks. The system includes monitoring for deep reorganizations and can alert operators when unusual chain reorganization patterns occur. However, the protocol itself does not include hard-coded checkpoints or finality gadgets that would compromise the permissionless nature of participation.

5.4 Security Analysis

The security of Nakamoto consensus has been extensively analyzed in academic literature. The fundamental security assumption is that honest miners control a majority of hashrate. Under this assumption, the probability that an attacker can successfully execute a double-spend decreases exponentially with confirmation depth.

Consider an attacker who broadcasts a payment, receives goods or services in exchange, and then attempts to reverse the payment by mining a private chain that excludes the transaction. The attacker must not only catch up to the honest chain but surpass it before the victim considers the transaction sufficiently confirmed. If the attacker controls a fraction q of total hashrate and the honest network controls fraction p = 1 - q, the probability that the attacker succeeds after the honest chain has extended by z blocks can be computed using a random walk model.

For realistic attacker hashrate fractions and moderate confirmation depths, success probability becomes negligible. For example, with an attacker controlling 30% of hashrate and a confirmation requirement of 20 blocks, the success probability is approximately 0.025%. This provides strong economic disincentives against double-spend attempts since the cost of acquiring and operating sufficient hashrate exceeds the potential gains from fraud for all but the largest transactions.

SPARK channels add an additional security consideration. Channel disputes are resolved by on-chain transactions during a timeout window. An attacker might attempt to prevent their counter-party from posting updates by controlling block production. However, the timeout window is measured in blocks rather than wall-clock time, and the protocol enforces that at most one UPDATE per channel per block can be accepted. With a 240-block timeout window and even if an attacker controls 40% of hashrate, they can expect to mine approximately 96 blocks during the window while the honest network mines 144 blocks. The counter-party has ample opportunity to post their highest sequence state.

5.5 Network Layer

The peer-to-peer networking layer uses libp2p, a modular networking stack supporting multiple transport protocols including TCP and QUIC. Nodes maintain connections to multiple peers across diverse network locations to resist eclipse attacks where an attacker isolates a victim from honest nodes. The protocol employs peer scoring mechanisms to identify and disconnect from misbehaving peers who send invalid data or exhibit unusual request patterns.

Block propagation uses compact blocks, where nodes announce new blocks by sending only the block header and transaction identifiers. Receiving nodes typically already have most transactions in their mempool from prior gossip, so they can reconstruct the full block without downloading transaction data again. This significantly reduces bandwidth consumption and improves propagation speed, which is critical for minimizing orphan rates.

Transaction propagation uses gossip protocols where nodes forward new transactions to their peers. To prevent denial-of-service attacks through transaction spam, nodes enforce rate limits on transaction acceptance and maintain per-peer quality scores. Peers who repeatedly send invalid transactions or violate rate limits are temporarily or permanently disconnected.

---

6. Economic Model

6.1 Monetary Policy

Winsdet implements a fixed supply cap of exactly one billion units, denominated as WNSD. Each WNSD can be subdivided to eight decimal places, yielding 100 billion billion indivisible

base units called WNSDi. All internal accounting operates in WNSDi to avoid floating-point arithmetic and ensure deterministic behavior.

The issuance schedule follows a halving model similar to Bitcoin but with parameters adjusted for the ten-second block interval. The initial block reward is 31.68808781 WNSD, which halves every 15,778,800 blocks. At a ten-second inter-block interval, each halving epoch lasts approximately five years. The reward continues halving until it reaches values too small to represent, at which point a deterministic cap-filling mechanism distributes the remaining supply.

Specifically, after standard halvings yield rewards smaller than one WNSDi, the protocol distributes exactly one WNSDi per block for the next 12,714,400 blocks. This cap-fill phase ensures the total supply reaches precisely 1,000,000,000.00000000 WNSD with no rounding errors or ambiguity. The issuance schedule is enforced through consensus rules and verified through unit tests that sum the total issuance across all epochs and confirm it equals the cap.

This deterministic approach eliminates any uncertainty about monetary supply. Unlike systems where supply depends on variable block production rates or discretionary adjustments, Winsdet's supply is fixed at the protocol level with mathematical precision. Market participants can verify the total supply by examining consensus code and running test suites, providing strong guarantees about scarcity.

6.2 Fee Market

Transaction fees provide two functions. During the issuance phase, fees supplement block rewards to incentivize mining. After issuance completes, fees become the sole mining incentive. Additionally, fees serve as an economic mechanism to allocate scarce block space among competing transactions.

The protocol enforces a minimum fee floor of one WNSDi per byte of transaction size. This prevents dust attacks where adversaries might spam the network with tiny zero-fee transactions. The minimum fee is intentionally low, imposing negligible cost on legitimate users while providing basic economic protection against abuse.

Users can increase transaction fees to prioritize confirmation during periods of high demand. The protocol includes Replace-by-Fee (RBF) mechanism allowing users to resubmit transactions with higher fees if initial submissions do not confirm promptly. RBF requires fee increases of at least 12.5% per replacement with a maximum of three replacements per transaction, preventing spam through repeated trivial replacements.

Package relay extends the fee model to dependent transactions. Some operations, particularly channel closures with fee sponsorship, require multiple related transactions to be mined together. Package relay allows nodes and miners to evaluate groups of dependent transactions by aggregate fee rate rather than considering each transaction independently. This prevents situations where a low-fee parent transaction blocks confirmation of a high-fee child transaction that depends on it.

6.3 Incentive Alignment

The economic model aligns incentives among various participants. Miners are incentivized to include high-fee transactions and maintain network security through block rewards and fees. The transition from inflationary rewards to fee-based incentives occurs gradually over multiple decades, allowing the ecosystem to adapt. By the time issuance completes, the system should have substantial transaction volume generating sufficient fee revenue to sustain mining operations.

Channel participants are incentivized to conduct transactions off-chain since channel payments incur no fees while on-chain payments do. However, they must periodically settle on-chain to establish channels and close them when relationships end. This creates a natural economic balance where routine transactions occur off-chain while channel lifecycle events generate on-chain fee revenue.

Full node operators receive no direct economic compensation, following the Bitcoin model where node operation is a public good provided by miners, merchants, exchanges, and interested parties who benefit from the network's continued operation. The system's light client support and modest resource requirements reduce barriers to running full nodes, improving decentralization through distributed participation rather than economic incentives.

---

7. Scalability and Performance

7.1 Transaction Throughput

Base layer throughput is intentionally constrained to maintain decentralization. With 128 kilobyte maximum block size, ten-second block intervals, and an average transaction size of approximately 200 bytes, the system supports roughly 6,400 transactions per block or 640 transactions per second in terms of raw settlement capacity. However, this figure represents theoretical maximum rather than typical operation. Policy guidance suggests 64 to 100 kilobyte target fills, yielding approximately 320 to 500 transactions per second under normal conditions.

This throughput is insufficient for serving as the sole payment layer for global commerce. However, it is adequate for a settlement layer where most activity occurs off-chain through SPARK channels. Consider a scenario with one million active channels. Each channel requires two on-chain transactions over its lifetime: one to open and one to close. If the average channel lifespan is 100 days, the system processes approximately 20,000 channel lifecycle transactions per day or 0.23 transactions per second, well within capacity. Meanwhile, these one million channels can collectively process millions of payments per second off-chain.

The key insight is that blockchain throughput measures settlement capacity rather than payment capacity. By moving payment activity off-chain while maintaining on-chain settlement, the

system achieves practical scalability without increasing block sizes or reducing block intervals that would compromise decentralization.

7.2 Storage Requirements

Storage growth follows from block production rate and average block size. With ten-second blocks producing 100 kilobyte blocks on average, the blockchain grows approximately 315 gigabytes per year in a worst-case scenario of continuous maximum capacity operation. In practice, actual growth will likely be lower since blocks will not consistently reach maximum fill.

However, nodes need not retain all historical data. The protocol supports body pruning where nodes discard transaction bodies after sufficient confirmations while retaining block headers and current state. Headers require only 116 bytes each, growing approximately 365 megabytes per year. Current state size depends on the number of active accounts and channels but remains bounded by practical economic limits. Even with millions of active accounts and channels, state size should remain under ten gigabytes.

Pruned nodes can validate new transactions and blocks without complete history since each block header commits to the current state through Merkle roots. New transactions are validated by checking that claimed inputs exist in current state and that state transitions follow consensus rules. The node need not know how those inputs were created since the state commitment provides cryptographic proof that they exist and contain sufficient value.

For users who require historical data, archival nodes can maintain complete blockchain history. However, the system does not depend on all nodes being archival. Even if only a small percentage of nodes maintain full history, they can serve historical data to other participants who need it occasionally, similar to how BitTorrent networks function with seeders and leechers.

7.3 Synchronization Time

New nodes must synchronize with the current blockchain state before participating in validation. Traditional synchronization requires downloading and validating every block from genesis, which can take days or weeks for large blockchains. Winsdet improves this through several mechanisms.

First, header-first synchronization allows nodes to download all block headers quickly, verifying proof-of-work and building the header chain. Since headers are small, this phase completes rapidly. Second, snapshot imports enable nodes to begin from a recent state checkpoint verified through publisher signatures rather than from genesis. Third, body pruning means nodes only need recent transaction bodies rather than complete history.

The combination of these techniques enables new nodes to achieve full validating capability within hours rather than days. A node can download the header chain, verify a recent snapshot, download bodies for the most recent blocks, and begin validating new blocks. This dramatically lowers the barrier to node operation while maintaining security guarantees.

7.4 Light Client Efficiency

Light clients achieve even better performance through receipt-based verification. A light client needs block headers and Merkle proofs for transactions of interest. For a user who receives ten payments per day, the client downloads ten receipts of perhaps one kilobyte each plus new block headers as they are produced. Total daily bandwidth is under 100 kilobytes, making mobile operation trivial even on constrained networks.

This efficiency enables genuine mobile wallet deployment without compromising security. The wallet can verify incoming payments immediately upon receiving receipts, provide strong security guarantees to the user, and operate on cellular networks without consuming significant data quotas. This stands in stark contrast to solutions that require mobile clients to trust remote servers to report balances and transaction histories accurately.

---

8. Privacy Considerations

8.1 Transaction Privacy

The protocol provides pseudonymity rather than anonymity. All transactions are publicly visible on the blockchain, but transactions are associated with addresses rather than real-world identities. Users who wish to maintain privacy can employ several techniques. Generating fresh addresses for each transaction prevents linkage across transactions. Using multiple addresses makes it difficult for observers to determine the full balance or activity of any single entity.

However, several mechanisms compromise privacy. Transactions with multiple inputs often reveal that all inputs belong to the same entity since coordination is required to create the transaction. Address reuse makes it trivial to track an entity's activity over time. Pattern analysis can identify common behaviors like exchange withdrawals or merchant payments, potentially enabling deanonymization.

The protocol does not include built-in privacy enhancements like Confidential Transactions or ring signatures. This represents a deliberate tradeoff. Privacy-enhancing technologies add complexity, increase transaction sizes, and may complicate regulatory compliance. The protocol prioritizes simplicity and auditability, leaving privacy enhancements to higher layers or external mixing services for users who require stronger privacy guarantees.

8.2 Channel Privacy

SPARK channels provide interesting privacy properties. While channel opening and closing are visible on-chain, the off-chain payments within channels are completely private from blockchain observers. Only the two channel participants know how many payments occurred or their

amounts. External observers see only the initial capacity allocation and final settlement balances.

This provides natural privacy for routine transactions without requiring special protocols. Two parties who transact frequently can open a channel once and conduct unlimited private transactions. Channel privacy is limited to the bilateral relationship, so network-wide privacy requires additional mechanisms, but the baseline privacy for participating parties is strong.

Channel rebalancing through CH_EXPAND operations reveals that a third party added capacity, but the specific payment history that led to the rebalancing need remains private. Similarly, dispute resolution through CH_CLOSE_UPDATE reveals that parties disagreed about closure terms but does not reveal the content of off-chain payments.

8.3 Network Privacy

Network-level privacy faces challenges in all blockchain systems. Nodes must announce transactions to peers for propagation, potentially revealing which node originated a transaction. While the peer-to-peer protocol does not explicitly link transactions to IP addresses, network observers can employ timing analysis and connection patterns to make statistical inferences about transaction origins.

Users concerned about network privacy should employ additional protections like Tor or VPNs to obscure their IP addresses. The protocol itself does not include built-in network privacy features since these are better addressed at lower network layers and would add complexity without providing complete solutions.

---

9. Operational Considerations

9.1 Node Operation

Running a full node requires modest resources. A modern consumer computer with a multi-core processor, 8 gigabytes of RAM, and 500 gigabytes of SSD storage suffices for operation as a validating node. Initial synchronization using snapshots completes within hours. Ongoing operation consumes minimal bandwidth under typical conditions, with brief spikes during new block propagation.

Operators should implement basic security practices including firewall rules to limit exposure, regular software updates, and backups of wallet private keys if the node maintains a wallet. The software includes health monitoring endpoints that expose metrics for operational dashboards. Operators can integrate these metrics with monitoring systems like Prometheus and Grafana to track node performance and detect anomalies.

Nodes can operate in several modes. Full archival nodes retain complete blockchain history and serve historical data to other participants. Pruned nodes maintain only recent history and current state, reducing storage requirements while retaining full validation capabilities. Mining nodes extend full or pruned nodes with mining capabilities, participating in block production. Light clients maintain only block headers and verify specific transactions through receipts.

9.2 Mining Operations

Mining follows standard proof-of-work patterns. Miners construct block templates by selecting transactions from the mempool, ordered by fee rate with package dependencies respected. They include a coinbase transaction that pays the block reward and collected fees to an address they control. They then iterate through nonce values attempting to find a header hash below the difficulty target.

Mining profitability depends on factors including hashrate, electricity costs, hardware efficiency, and network difficulty. The protocol itself remains neutral about mining centralization versus decentralization. Market forces determine optimal mining configurations. However, the ten-second block interval and 128 kilobyte maximum block size keep resource requirements for full validation low, enabling more participants to verify miner behavior even if mining itself becomes specialized.

Mining pools can form to reduce variance in mining returns, similar to Bitcoin mining pools. The protocol does not include built-in pool support, but standard pool protocols can operate atop the base layer. Pool participants contribute hashrate to the pool operator who distributes rewards according to contributed work.

9.3 Exchange Integration

Cryptocurrency exchanges face specific challenges when integrating new blockchain systems. Winsdet provides several features that simplify exchange integration. The PAYM transaction type enables efficient batch withdrawals where the exchange can combine multiple customer withdrawals into a single transaction. This reduces transaction fees and blockchain space consumption compared to individual withdrawals.

Receipt-based verification enables exchanges to confirm customer deposits without running full nodes, though most exchanges will likely operate full nodes for maximum security and control. The deterministic transaction ID computation prevents malleability attacks where attackers might modify transaction formats to change identifiers while preserving validity.

The protocol includes a confirmation maturity period for coinbase transactions, preventing miners from spending newly mined coins for 100 blocks. This prevents situations where a blockchain reorganization might invalidate coinbase transactions that were already spent, creating accounting difficulties. Regular transactions do not have maturity requirements, though exchanges typically impose their own confirmation requirements based on transaction value and perceived risk.

9.4 Merchant Adoption

Merchants accepting cryptocurrency payments must balance transaction irreversibility against confirmation latency. For low-value purchases, merchants might accept unconfirmed transactions, relying on the difficulty of executing double-spend attacks against retail transactions. For moderate values, six confirmations provide strong security since the cost of acquiring sufficient hashrate to reverse six-block-deep transactions exceeds the payment value for routine purchases. High-value transactions can require more confirmations proportional to the value at stake.

SPARK channels provide an alternative for repeat customers. A customer establishes a channel with the merchant through an initial on-chain transaction. Subsequent purchases occur instantly off-chain through channel payments. When the business relationship ends, the channel closes with a final settlement transaction. This model works particularly well for subscriptions, regular services, or businesses with recurring customer relationships.

The fee sponsorship mechanism proves valuable for merchants since they can guarantee that customer channels can close even if customers lack funds for fees. The merchant sponsors closure fees through the FeeEnv structure, ensuring channels do not become stuck due to fee market volatility. This removes friction from the customer experience while giving merchants control over when and how channels close.

---

10. Future Directions

10.1 Protocol Extensions

The base protocol intentionally maintains a minimal feature set to reduce complexity and security risks. However, several directions for future development exist that could enhance functionality without compromising core properties. Cross-channel atomic swaps would enable more sophisticated payment routing where value can be moved between channels atomically. Channel factories, which are channels that can spawn sub-channels, could improve capital efficiency for hubs serving many clients.

Probabilistic payments might enable micropayments smaller than the base unit by using cryptographic lottery mechanisms. Confidential transactions could enhance privacy by hiding payment amounts while maintaining verifiability. These and other enhancements can be considered for future protocol versions based on community demand and security analysis.

10.2 Ecosystem Development

A healthy blockchain ecosystem requires more than just protocol development. Wallet software must be user-friendly and secure. Block explorers should provide transparency into on-chain

activity. Developer tools and libraries enable application development. Education resources help users understand the technology and its implications.

The Winsdet development team anticipates that third parties will create many of these ecosystem components. The protocol itself remains neutral and non-proprietary, enabling anyone to build compatible software and services. Open-source reference implementations provide starting points for developers while permitting alternative implementations with different design tradeoffs.

10.3 Research Questions

Several open research questions deserve investigation. Optimal channel topology remains unclear. Should the network form around hub-and-spoke models with professional routing nodes, or distributed mesh networks where everyone routes payments? How do privacy enhancements affect blockchain analysis and regulatory compliance? Can multi-party channels or channel virtualization improve capital efficiency without compromising security?

Academic researchers are encouraged to investigate these questions. The protocol's open design enables experimentation through testnet deployments without risking mainnet security. Research findings can inform future protocol development and best practices for application developers.

---

11. Conclusion

We have presented Winsdet, a blockchain protocol that achieves scalability through architectural separation of settlement and payment layers. By integrating SPARK payment channels directly into consensus rules, the system provides deterministic guarantees about channel behavior while maintaining the security properties of the base layer. Light client verification through cryptographic receipts enables resource-constrained participants to verify blockchain state without downloading complete transaction history, significantly improving accessibility.

The protocol makes deliberate tradeoffs. On-chain throughput is intentionally limited to maintain decentralization, accepting that most payment activity occurs off-chain. Privacy is provided through pseudonymity rather than cryptographic anonymity, prioritizing transparency and auditability. The monetary policy is completely deterministic with no discretionary adjustments, providing certainty about scarcity but eliminating flexibility to respond to economic conditions.

These design decisions reflect a philosophy that blockchain systems should be optimized for settlement rather than attempting to serve all use cases directly. By focusing on what blockchain systems do well—providing cryptographic proof of transaction ordering and state

commitment—the protocol enables higher-layer systems to add additional functionality without compromising base layer security.

Winsdet builds upon fifteen years of blockchain development, incorporating lessons learned from Bitcoin, Lightning Network, and numerous other systems. It represents not a revolution but an evolution, taking proven concepts and integrating them more tightly to reduce complexity and improve security. The result is a system that maintains the permissionless, trustless properties that make blockchain technology valuable while achieving the performance characteristics necessary for practical deployment.

The protocol is operational and available for public use. Development will continue through an open process where improvements are proposed, analyzed, and implemented through community consensus. We invite researchers, developers, and users to examine the system, identify shortcomings, and contribute to its evolution. The goal is not merely to create another cryptocurrency but to demonstrate that blockchain scalability can be achieved without sacrificing the properties that make these systems valuable in the first place.

References

[1] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.

[2] Poon, J., & Dryja, T. (2016). The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments.

[3] Merkle, R. C. (1980). Protocols for Public Key Cryptosystems. Proceedings of the 1980 IEEE Symposium on Security and Privacy.

[4] O'Connor, J., Aumasson, J. P., Neves, S., & Wilcox-O'Hearn, Z. (2020). BLAKE3: One Function, Fast Everywhere.

[5] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., & Yang, B. Y. (2012). High-speed high-security signatures. Journal of Cryptographic Engineering.

[6] Bormann, C., & Hoffman, P. (2020). Concise Binary Object Representation (CBOR). RFC 8949.

[7] Zawy (2017). Linearly Weighted Moving Average (LWMA) difficulty algorithm. GitHub repository: zawy12/difficulty-algorithms.

[8] Decker, C., & Wattenhofer, R. (2015). A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels. Symposium on Self-Stabilizing Systems.

[9] Miller, A., et al. (2017). Sprites and State Channels: Payment Networks that Go Faster than Lightning. Financial Cryptography and Data Security.

[10] Bünz, B., Agrawal, S., Zamani, M., & Boneh, D. (2020). Zether: Towards Privacy in a Smart Contract World. Financial Cryptography and Data Security.

End of Document

Winsdet Development Team
October 15, 2025