Arch: The Permissionless Financial Rails for a Bitcoin-denominated World

Abstract. Arch enhances Bitcoin's capabilities by enabling fast, secure, and fully-verifiable smart contracts without requiring users to bridge their assets. It does so without a soft fork, instead leveraging a novel infrastructure stack that combines a decentralized validator network, a UTXO-aware execution environment based on the extended Berkeley Packet Filter (eBPF) virtual machine, and cryptographic multisig using FROST+ROAST signature schemes. Unlike L2s and meta-protocols where users must bridge their assets before interacting with smart contracts, Arch allows users to send their assets directly to smart contracts using native UTXO-based Bitcoin transactions. This design offers more security and utility than L2s that fracture liquidity and add bridging risks. It also allows for greater interoperability, unlocking the Bitcoin economy by connecting emerging decentralized BTC-native apps, from DeFi and gaming to prediction markets and social networks.

1 The Bitcoin Builder's Dilemma

Since its whitepaper debut in 2008, Bitcoin has emerged as the canonical gold standard in the digital assets sector yet has failed to capture the "code is law" developer market due to its heavy scripting limitations and overall lack of expressivity. Centralized solutions have proven to be nothing more than a flash in the pan and have oftentimes resulted in extreme financial loss due to hacks and mismanagement.

More secure protocol extensions such as Omni Layer, Colored Coins, and RGB had brief moments of adoption yet failed to scale and forced users away from their habitual Bitcoin access point–their wallet. The emergence of metaprotocols like Ordinals and Runes have emerged as the best standard to tokenize assets on Bitcoin, yet continue to struggle due to Bitcoin's slow block times and lack of expressivity.

Developers know that the next trillion dollar opportunity is unlocking DeFi and capital formation on pristine Bitcoin but have had no way to access that market. Those seeking programmability are forced to wrap or bridge BTC assets onto more expressive blockchains like Solana or Ethereum, fracturing liquidity and introducing centralized custody risks in the process.

This is the "Bitcoin Builder's Dilemma" that Arch was built to solve.

1.1 Limited Scripting and Expressivity Bitcoin's Script language was intentionally designed with severe limitations to prioritize security and determinism. While this approach served Bitcoin's primary function as hard money, it came at the cost of expressivity. Developers face a restrictive environment where even basic programming constructs like loops are unavailable, making it nearly impossible to implement complex financial applications directly on Bitcoin (e.g. AMMs, pool-based lending, etc.).

1.2 Poor UX and Fractured Liquidity Bitcoin's user experience challenges present significant hurdles for developers building interactive applications. The network's fundamental design, with 10-minute block times, creates inherent latency issues that make responsive applications difficult to achieve. When users expect near-instant feedback, waiting minutes or hours for transaction confirmations creates friction that drives them away. Alternative scaling solutions attempt to address these limitations but introduce their own complexities—requiring users to bridge assets to completely different ecosystems with separate wallets, security models, and interfaces. Even these solutions typically offer only incremental improvements with multi-second finality times, rather than the sub-second responsiveness users have come to expect from modern applications. This fragmentation of liquidity and attention across disconnected environments ultimately limits the growth potential of any Bitcoin-based application ecosystem.

2 Why L2s and Metaprotocols Aren't Enough

Bitcoin Layer 2 solutions try to address the issues of Bitcoin by building their own execution layers, which users must bridge their assets onto. This process allows for scalable programmability, often showcasing much faster block times and larger block sizes. However, bridging their assets transfers trust away from the Bitcoin base layer — by far the most liquid, secure, and decentralized blockchain — to these L2s.

This isn't appealing to many Bitcoiners, who have already had the ability to access programmability by bridging to Ethereum for years yet have largely chosen not to do so. The total value locked of "wrapped" BTC assets in DeFi — Bitcoin that is bridged onto Ethereum — is about $10 billion as of writing, less than 1% of the total Bitcoin market cap.

A few emerging meta-protocols have also tried to add new capabilities to Bitcoin by increasing off-chain compute capacity. However, they can only handle state changes. They can't do asset transfers, which require the ability to sign transactions programmatically and in a decentralized, secure fashion on the Bitcoin base layer.

In short, they have the same performance limitations of the Bitcoin L1 without the utility of Bitcoin L2s. Plus, most also require that users bridge their assets.

Assets that are bridged onto L2s and meta-protocols are siloed. They are unable to communicate with each other or the base layer. This means they lack the necessary interoperability components for multi-party contracts and cross-program invocation as seen on other decentralized networks.

This fragmentation lowers addressable liquidity, as different chains and meta-protocols cannot easily inter-operate on Bitcoin, making it harder to aggregate the types of deep liquidity sources needed to operate more mature DeFi applications.

To build decentralized utility and programmability, Bitcoin needs a truly bridge-free execution platform that is Turing-complete, meaning it is capable of executing both state changes and asset transfers directly on the base layer.

It must be able to power the complex, interoperable smart contract functionalities needed to support decentralized apps without fracturing liquidity or compromising security. And, critically, it must do so in a trust-minimized way that doesn't force Bitcoiners to trust their assets with bridges or insufficiently secure protocols.

3 The Arch Unlock

Arch expands what's possible in the Bitcoin ecosystem by providing a user and developer experience previously unattainable on Bitcoin. As the programmability layer that Bitcoin has been missing, Arch preserves the monetary integrity of pristine BTC while enabling an open, fast, and expressive economy.

3.1 Seamless Bitcoin Wallet Integration Unlike other solutions that force users away from their Bitcoin wallet, Arch Network's novel Taproot address compatibility account model allows users to access Arch apps directly from their existing Bitcoin wallets:

Native compatibility: Use familiar Bitcoin wallets like Xverse, Unisat, MagicEden, and Ledger to interact with Arch applications.

No bridging required: Access Arch functionality without transferring assets to external chains or secondary wallets that introduce additional trust assumptions.

Unified asset management: Manage all Bitcoin assets (BTC, Ordinals, Runes) in a single wallet interface.

Interoperability: Move seamlessly between Bitcoin applications and Arch-enhanced apps.

This deep integration with the Bitcoin wallet ecosystem eliminates the fragmentation of liquidity and attention that has plagued previous Bitcoin extension attempts. Arch enables a unified experience by preserving the security model users trust while expanding what they can do with their Bitcoin assets.

3.2 Flexible Trust Models for Developers Arch's unique approach gives developers unprecedented flexibility in how they balance Bitcoin's security with modern application performance requirements:

Customizable security models: Choose the optimal point on the trust spectrum for each application or transaction type, breaking free from Bitcoin's security vs. innovation dilemma.

Direct Bitcoin settlement: Process transactions directly on Bitcoin's L1 for maximum security when needed using Arch's high threshold signature scheme.

Pre-confirmation acceleration: Utilize Arch's pre-confirmation system to achieve sub-second responsiveness, overcoming Bitcoin's inherent 10-minute block times.

Solana-like UX: Deliver the smooth, instantaneous experiences users expect from modern applications without sacrificing Bitcoin's security guarantees.

This flexibility allows developers to build applications that are both responsive and secure, without sacrificing either quality. Through Arch's UTXO-aware execution environment based on the eBPF virtual machine with custom syscalls, developers can finally implement complex financial applications interoperably with Bitcoin.

3.3 Supercharged Bitcoin Assets While metaprotocols like Runes and BRC-20 have emerged as significant standards to tokenize assets on Bitcoin, they continue to struggle due to Bitcoin's slow block times and lack of expressivity. Arch has built infrastructure to optimize and make Bitcoin assets more frictionless and fungible.

UTXO optimization: Specialized accounts model designed specifically for the complexities of UTXO asset organization.

State sharding: Using sharding techniques to enhance throughput for state-dependent transactions.

Real-time mempool indexing: Index Bitcoin, Ordinals, and Runes transactions in the mempool at significantly higher volumes than previously possible.

Reliable block inclusion: Leveraging robust fee estimation, state chaining, and non-standard Bitcoin transactions, we provide developers with high probability block inclusion

Improved liquidity mechanisms: Allow efficient trading pools and marketplaces for Bitcoin-native assets, unlocking DeFi and capital formation on pristine Bitcoin

Powered by Arch's decentralized validator network that facilitates the seamless coordination of Bitcoin-native transactions via stake-weighted dPoS consensus, Arch transforms how Bitcoin assets function. This enables them to scale effectively while maintaining the decentralized properties that make them valuable.

4 Architecture Overview

Arch combines the power of Bitcoin with the speed and execution of Solana. The network is composed of a validator set running a dPoS consensus on top of our novel ArchVM combined with a cutting edge cryptographic multisig — one that uses FROST and ROAST threshold signature schemes that settle directly on Bitcoin.

4.1 Consensus Arch's consensus is powered by Arch's native token using a dPoS model for block production and threshold key distribution. The state and activity of the network are propagated across nodes through the GossipSub protocol, ensuring that all validators receive updates in a decentralized and fault-tolerant manner. Leveraging FROST + ROAST, Arch comes to consensus on new user-submitted Bitcoin UTXOs and settles them directly on Bitcoin with our cryptographic multi-sig architecture.

Block Production The block production process in our dPoS consensus begins with leader selection, which operates on a predetermined schedule for each epoch or fixed time period. Leaders are strategically selected based on their stake weight in the network, creating a deterministic schedule that is transparent and known to all participating validators. To maintain system reliability, multiple backup leaders are designated for each slot, providing essential fault tolerance should the primary leader fail to produce a block within the allocated time frame.

When a validator assumes the leadership role, they immediately begin the transaction collection and verification phase. During this critical stage, the leader gathers all pending transactions from the mempool—the network's temporary storage for unconfirmed transactions. Each transaction undergoes rigorous verification, with the leader checking signature validity and ensuring all transactions adhere

to the protocol's rules. Once verified, transactions are ordered based on first in, first out, enabling fair inclusion in the upcoming block.

The final stage involves block formation, where the leader assembles a comprehensive block structure that serves as an immutable record in the blockchain. This structure contains several essential components: a reference to the previous block to maintain chain continuity, a precise timestamp documenting when the block was created, and a list of transaction IDs that were included in the block.

Consensus Process When validators receive a new block, they engage in a comprehensive validation process to ensure network integrity. First, they verify that the block producer is indeed the designated leader according to the predetermined schedule, preventing unauthorized block creation. Next, validators meticulously validate all transaction signatures within the block to confirm their authenticity and prevent forgery. They then execute each transaction and verify the resulting UTXO states, ensuring that all state transitions follow protocol rules and maintain data consistency across the network. Finally, validators perform thorough checks for any consensus rule violations, examining block structure, timestamp validity, transaction ordering, and other protocol-specific requirements before accepting the block as valid and adding it to their local chain.

4.2 Understanding FROST FROST (Flexible Round-Optimized Schnorr Threshold) is an advanced threshold signature scheme that enables multiple parties to collectively generate a single valid Schnorr signature while minimizing communication rounds. Its key innovation is achieving single-round signing under ideal conditions. FROST maintains the threshold property where only a specified subset of participants (t out of n) need to cooperate to produce a valid signature, allowing flexibility when some participants are offline. While prioritizing efficiency over robustness (all participating signers must be honest), FROST incorporates security mechanisms to detect and mitigate forgery attempts by malicious participants. The scheme supports both two-round and optimized single-round signing protocols to accommodate various operational requirements, making it suitable for applications ranging from cryptocurrency transaction signing to secure multi-party consensus in distributed networks.

Read more about FROST.

4.3 Understanding ROAST ROAST (Robust Asynchronous Schnorr Threshold signatures) is an enhancement protocol designed to transform semi-interactive threshold signature schemes into robust and asynchronous signing protocols. It addresses critical limitations in existing Schnorr threshold signature schemes that often fail in real-world applications due to assumptions about synchronous network conditions and lack of robustness against disruptive participants. ROAST achieves robustness by orchestrating multiple concurrent instances of the underlying signing protocol with different subsets of signers, ensuring that even if some sessions fail due to malicious behavior or network issues, a valid signature can still be generated when a threshold of honest signers participate. Key requirements for ROAST implementation include support for identifiable aborts, resistance to forgery under concurrent signing sessions, and a semi-interactive structure with one preprocessing and one signing round. By including mechanisms to identify and exclude disruptive signers without restarting the entire process, ROAST significantly advances the practical deployment of threshold signatures in distributed systems.

Read more about ROAST.

5 Execution

The Arch Network connects the security and immutability of Bitcoin with the programmability and flexibility of modern layer-2 networks. At the heart of this innovation is the ArchVM, a specialized virtual machine designed to execute complex smart contract logic while maintaining Bitcoin state consistency.

The following sections explore the architecture of the ArchVM, its execution model, and the unique mechanisms that enable it to maintain state integrity across both Arch Network and the Bitcoin blockchain.

5.1 ArchVM and Runtime The ArchVM uses a fork of Solana's eBPF (extended Berkeley Packet Filter) virtual machine, providing a secure and performant environment for executing smart contract programs. While using familiar terminology to Solana, ArchVM is uniquely able to interact directly with Bitcoin's UTXO model.

Foundation: Rust-Based eBPF Virtual Machine Like Solana's implementation, the ArchVM executes programs compiled to eBPF format, uses the Solana BPF compiler (SBF compiler) to transform Rust code into ELF binary, and provides a restricted instruction set designed for safe execution.

This foundation ensures that developers familiar with Solana's programming model can easily build on Arch while benefiting from its Bitcoin integration capabilities.

Key Components The ArchVM architecture consists of several critical components working together:

1. **Program Registry**: Maintains the catalog of deployed programs on Arch Network and manages their lifecycle.
2. **Instruction Processor**: Executes individual instructions within transactions according to the eBPF instruction set.
3. **Memory Management**: Controls program memory access, provides sandboxing, and enforces memory safety constraints.
4. **Syscall Interface**: Provides a gateway to system-level functionality, including both standard and Bitcoin-specific operations.
5. **Compute Meter**: Tracks and limits computational resources to prevent excessive resource consumption
6. **DAG Transaction Processor**: Organizes transactions in a directed acyclic graph to track dependencies and execution order.

5.2 Extended Syscalls The most significant innovation in the ArchVM is the addition of Bitcoin-specific syscalls that allow smart contracts to interact with the Bitcoin blockchain:

- UTXO Transaction Processing: Syscalls to create, sign, and verify Bitcoin UTXO transactions
- Bitcoin Script Execution: Ability to verify and create Bitcoin scripts
- Direct Bitcoin State Access: Read Bitcoin blockchain state and UTXO information
- Transaction Posting: Submit transactions directly to the Bitcoin network

These syscalls serve as the bridge between the programmable environment of Arch and the fundamental UTXO model of Bitcoin.

5.3 Cryptographic Multisig Arch introduces a programmable multisig with a dynamic validator set that allows signers to enter and exit the network in a decentralized way based on stake weight and performance.

By combining FROST + ROAST with a Proof-of-Stake economic security model, Arch enables:

- Validator sets that are dynamic and can change over time
- Permissionless participation in securing the network by staking ARCH
- Multisig control that becomes progressively more decentralized and censorship-resistant

5.4 DAG-Based Transaction Dependency Graph Unlike traditional blockchain systems with linear transaction processing, Arch Network employs a transaction specific Directed Acyclic Graph (DAG) structure. This approach enables more efficient parallelization, precise dependency tracking, and sophisticated state management when interacting with Bitcoin.

The transaction graph consists of individual nodes (transactions) in the Arch network and edges (the dependencies between transactions, representation transactions that operate on the same Arch accounts).

Each transaction in the graph is represented by a structure that maintains which transactions were processed first (called "previous transactions") and which are dependent on the current transaction (called "Next transactions").

The graph also tracks the current node state, confirming whether a transaction is anchored on Bitcoin or state-only — that is, Arch only, with no Bitcoin anchoring — and also if the transaction has been rolled back or not.

This graph structure allows the system to maintain a complete picture of transaction relationships, which is crucial for handling complex potential state inconsistencies.

When a transaction fails, the system can precisely identify which other transactions are affected, allowing Arch to rollback only affected transactions.

5.5 Rollback/Reapply Mechanism When Bitcoin state changes affect Arch transactions:

1. The DAG is traversed to identify affected transactions
2. Affected branches of the DAG are rolled back
3. Transactions incompatible with the new Bitcoin state are failed
4. Previously failed transactions that are now compatible are reapplied
5. The DAG is reconstructed with the updated state

5.6 Enabling Bitcoin Pre-Confirmations The DAG-based Transaction Dependency Graph and Rollback/Reapply Mechanism enable Arch to implement pre-confirmations to overcome Bitcoin's inherent slowness without compromising security.

Consensus and Finality

Once consensus is reached on a block, all transactions in the DAG that are part of that block are preconfirmed on Arch

1. Selected transactions are anchored to Bitcoin according to their requirements
2. The indexer monitors Bitcoin for confirmations and reorganizations
3. The rollback/reapply mechanism maintains state consistency if Bitcoin state changes

Instead of waiting for full Bitcoin block confirmations —which create poor user experiences and limit DeFi liquidity flow — Arch provides "soft-confirmations" that allow applications to proceed immediately as if transactions will be included in the next Bitcoin block.

By chaining state transitions together, Arch guarantees state consistency between Bitcoin and Arch. The rollback mechanism can reverse state transitions on Arch if a transaction gets evicted from the mempool or experiences a reorg on Bitcoin.

Crucially, Arch can perform these rollbacks with single-transaction granularity, meaning only affected transactions need reversal rather than the entire network, maintaining system stability while enabling much faster transaction processing.

6 Transaction Lifecycle

Users submit program transaction requests through an RPC interface, providing the associated Bitcoin state anchors, input data, and transaction fees (paid in BTC). Each node of the Decentralized Verifier Network distributes the request within the network, runs the program request, signs off on the result, and shares the signed result with the elected leader node. As soon as a threshold of signatures has been collected, the leader node submits the resulting Bitcoin transaction.

6.1 Program Execution Process The execution of an Arch program follows these steps:

1. Program Deployment:
   - The program's eBPF code is verified against the ISA constraints
   - The program is stored in an executable account
2. Transaction Preparation:
   - Transaction instructions targeting the program are analyzed
   - Dependencies are identified and placed in the DAG
3. Execution Context Initialization:
   - Parameter serialization (program ID, account info, instruction data)
   - Stack and heap allocation

   - Memory mapping configuration
   - Syscall context setup
4. Program Execution:
   - The eBPF program is executed using JIT compilation or interpretation
   - Bitcoin-specific syscalls are processed when encountered
   - Compute units are tracked to prevent infinite loops or DoS attacks
5. Post-Execution Bitcoin Validation:
   - Titan indexer validates Bitcoin transactions referenced by the program
   - State is updated based on Bitcoin blockchain confirmation
6. Reorg Handling:
   - If Bitcoin reorgs or fee replacements are detected, affected transactions are identified
   - The DAG is traversed to find dependent transactions
   - Impacted transactions are rolled back or reapplied as needed

7 Conclusion

Arch is building the permissionless financial rails for a Bitcoin-denominated world. By solving the Bitcoin Builder's Dilemma without a soft fork, Arch unlocks a user & developer experience that previously wasn't possible on Bitcoin. With its novel infrastructure stack that combines a decentralized validator network, a UTXO-aware execution environment, and cryptographic multisig, Arch preserves the monetary integrity of pristine BTC while enabling an open, fast, and expressive economy around it. This creates new opportunities for innovation and adoption in the Bitcoin ecosystem, from DeFi and GameFi to Art and Metaprotocols.