Nubit DA: Bitcoin-Native Data Availability Layer with Instant Finality

The Nubit Team
Riema Labs
nubit@riema.xyz

ABSTRACT

The rapid advancement of blockchain technology, with Bitcoin at its core, has significantly influenced the digital economy. However, Bitcoin's growth has been hindered by challenges in scaling and ensuring efficient transaction validation and finality, especially with the increasing demand for Bitcoin inscriptions and layer-2 solutions. A critical issue in this context is the data availability problem, which limits the network's capacity to validate and finalize transactions swiftly and efficiently. To address this, we introduce *Nubit DA*, a pioneering Bitcoin-native data availability layer designed to achieve instant finality, thereby considerably enhancing transaction confirmation times. *Nubit DA* incorporates a novel consensus algorithm, named as NubitBFT, which supports a vast network of validators, ensuring the robust security and censorship resistance inherent to the Bitcoin blockchain. Addressing the scalability challenges, *Nubit DA* employs a signature scheme based on Zero-Knowledge Proofs (ZKP) to minimize communication overhead among validators, enabling the network to scale effectively without compromising security. Additionally, it introduces an innovative approach to data availability through data availability sampling and a decentralized Bitcoin payment scheme, further enhancing the efficiency and trustlessness of the Bitcoin ecosystem. This paper details the design and implementation of *Nubit DA*, highlighting its contributions to achieving a scalable, secure, and efficient blockchain network. Through these advancements, *Nubit DA* not only addresses pressing scalability and data availability challenges but also significantly improves the Bitcoin user experience by reducing latency and facilitating a more efficient payment protocol.

1 INTRODUCTION

The continuous evolution of blockchain technology has been instrumental in shaping the digital economy, with Bitcoin standing at the forefront as the pioneering decentralized digital currency. Since its inception, Bitcoin has faced numerous challenges, particularly in scaling its network to support higher transaction throughput and faster finality while maintaining its core principles of decentralization and security. Among these challenges, the data availability problem poses a significant bottleneck, impacting the network's ability to efficiently validate and finalize transactions. The recent surging demand for Bitcoin inscriptions and Bitcoin layer-2 further amplifies this problem.

A data availability layer [2] refers to a specialized infrastructure or protocol layer designed to ensure that data necessary for validating transactions and blocks is readily accessible to all participants in the network. This concept is particularly relevant in the design of scalable blockchain architectures, where managing the sheer volume of data efficiently without compromising security or decentralization becomes a significant challenge. By separating data handling from transaction processing, a data availability layer aims to enhance scalability (by enabling more transactions per second) without sacrificing decentralization.

The goal: We propose *Nubit DA*, a Bitcoin-native data availability layer with instant finality, which refers to the ability of a blockchain network to confirm transactions within a single block confirmation, drastically reducing the time it takes for transactions to be considered final and irrevocable. In particular, *Nubit DA*'s novel consensus algorithm supports a large volume of validators while inheriting the tamper-resistance and censorship resistance of Bitcoin blockchain.

Challenges: However, developing a data availability layer for the Bitcoin ecosystem is quite challenging due to the following two reasons:

- *Validator Size.* The consensus mechanism ensures that all participants in the network agree on a single source of truth, even in the presence of malicious actors. However, as the number of validators grows, the limitations of its consensus algorithm in terms of scalability, communication overhead, and speed have become increasingly apparent.
- *Block Size.* As the number and complexity of Bitcoin applications grow, the average block space also increases. In that case, it will be challenging for user clients to efficiently verify data availability.

Our solution: To address the first challenge, *Nubit DA* will adopt a signature aggregation scheme based on zkSNARKs, a cryptographic primitive that allows one party to prove to another that a statement is true, without revealing any additional information apart from the fact that the statement is indeed true. In particular, in the Tendermint [10] scheme, the complexity of each round of communication is $O(n^2)$. Assuming there are N validators and a block hash H, a proof π can demonstrate that H has been signed by at least k validators. This proof will only be accepted when it is verified by a validator and $k \geq 2/3N$. This technology, named as *NubitBFT*, will significantly reduce the communication volume between validators in verifying the validity of signatures from $O(n^2)$ to $O(n \log n)$, verifying the correctness of transactions without having to access the entire dataset, thereby significantly reducing the computational overhead and increasing the scalability of the network.

To address the second challenge, traditional blockchain schemes, such as Bitcoin, require users to download all data to finally verify data availability, which is intolerable for a high-performance blockchain: this means that any ordinary user would need TB-level storage to trust the network. To mitigate this challenge, *Nubit DA* incorporates Data Availability Sampling (DAS), a solution proposed in [3]. Furthermore, to reduce the size of data communication among validators, *Nubit DA* introduces a technique called *Block Dispersion*. This method enables each validator to retain a faithful yet minimal portion of the original data, eliminating the need for

downloading all data from the leader, while the entire network still provides a complete replica of the data.

Contributions. In summary, this paper makes the following contributions:

- We design and implement *Nubit DA*, the first Bitcoin-native data availability layer.
- We design and implement an innovative, high-performance consensus protocol that achieves *Instant Finality* without compromising security.
- Our novel signature aggregation scheme, powered by advanced zero-knowledge proofs, can dramatically reduce *Nubit DA*'s communication costs.

2 BACKGROUND

In this section, we provide some necessary background on blockchain, data availability, and zero-knowledge proofs.

2.1 Blockchain

In recent years, blockchain technology, fundamentally a distributed ledger, has emerged as a trusted paradigm for asset issuance and transaction processing due to its resistance to censorship, tamper-proof, and decentralization. Technically, a blockchain essentially comprises three core components: the consensus layer, the settlement layer, and the data layer. The consensus layer is responsible for reaching an agreement on transactions such as proposed blocks, verifying the validity of blocks, and penalizing dishonest nodes among the network's participants. The settlement layer maintains the integrity of the blockchain ledger, while the data layer stores and maintains the data on the blockchain. The tamper-proof of a blockchain is based on the assumption that the economic cost of tampering with the network exceeds the total value of assets issued on it. Correspondingly, two types of consensus models are used to establish the network's tamper-proof: one is the Proof of Stake (PoS) model, where network participants "purchase" the possibility of being selected to propose a block and participate in its validation by staking a certain value of tokens, with the tamper-proof determined by the total value of the staked tokens. The other is the Proof of Work (PoW) model, where participants propose blocks by solving specific puzzles and submitting proof of their work, which can be verified by other nodes, with the tamper-proof determined by the overall computational power of the network, indirectly anchoring it to real economic value. Based on these fundamental models, to provide different characteristics, many blockchain paradigms have been established, with Bitcoin and Ethereum being representatives of the PoS and PoW networks, respectively.

Bitcoin. Bitcoin [9] is the first modern blockchain system. As of 2024, due to its PoW consensus model and a network hash rate exceeding 500M Trillion Hash/s, the Bitcoin blockchain is considered to have the best tamper-proof and censorship-resist among blockchain networks. Bitcoin itself has data storage capabilities, but due to its protocol's limitation on block space (1MB), the cost of storing data on Bitcoin is very expensive in practice. Additionally, Bitcoin's settlement layer lacks support for Turing-complete languages, so any derivative protocols relying on Bitcoin settlement require more specialized designs due to this limitation. The Lightning Network is an example of a protocol built on top of Bitcoin, designed to facilitate fast, small-scale BTC payments between users.

Ethereum. Ethereum [26] is the most popular network within the PoS consensus model: it optimizes its consensus protocol to allow for an extremely large number of validators (around 900,000 as of 2024), thus considered to have a high degree of decentralization and resistance to censorship. However, as a trade-off, Ethereum theoretically requires about 15 minutes to confirm the finality of a block. The cost of data storage on Ethereum's data layer remains high, prompting developers to consider low-cost data availability[3]. Compared to Bitcoin, Ethereum's settlement layer possesses a Turing-complete virtual machine responsible for executing user-submitted code, thereby fostering a more prosperous ecosystem.

2.2 Data Availability

In the context of blockchain, data availability refers to the assurance by network participants that the data uploaded by users is faithfully proposed and stored in certain nodes of the network, and that these nodes can prove this property to other participants. To achieve consensus on data availability, the most straightforward approach, such as methods in early blockchain systems like Bitcoin, requires network participants to download all block data and verify its validity, leading to significant scalability issues: if the blockchain stores large enough data, ordinary users may not be able to afford the cost of processing this data or may not wish to spend a lot of bandwidth on data they are not interested in. As a result, they may be unable or unwilling to participate in verification, leading to centralization issues in the network. To mitigate this challenge, LazyLedger [2] proposed a new technique that uses random sampling to verify data availability. Technically, Data Availability Sampling (DAS) allows network participants to ensure the full availability of block data without requiring any participant to download it completely. This scheme allows potentially malicious block proposers to encode the content of the block into a commitment σ and a complete encoded block π. The commitment σ is added to the block header and it allows light nodes, which are run by ordinary network participants, to verify the availability of the complete π by requesting a few positions in π randomly. If a sufficient number of light nodes successfully probe π, DAS ensures that the data is fully available. Note that a single light node cannot be certain of the full availability of the data, as it queries only a small portion of the encoded data, therefore DAS still requires a sufficiently large group of light nodes.

2.3 Zero-knowledge proofs

A zero-knowledge proof (ZKP) enables one party, the *prover* $\mathcal{P}$, to demonstrate to another party, the *verifier* $\mathcal{V}$, that they possess certain secret information without actually disclosing the information itself. This capability is crucial in various applications, particularly within the realm of blockchain technology. In the context of data availability, ZKPs could effectively verify the correctness of transactions without accessing the entire dataset, thus dramatically reducing the communication overhead among validators.

To elaborate, consider a computational process C that utilizes public inputs x and secret inputs y. The prover's objective is to assure the verifier that a specific output z is the genuine result of

$C(x, y)$, all while maintaining the confidentiality of y. The essence of a ZKP lies in its ability to confirm the prover's access to y without disclosing any information about y itself. Among the diverse array of zero-knowledge protocols, *zkSNARKs* (*Zero-Knowledge Succinct Non-interactive ARguments of Knowledge*) have garnered attention for their concise proof sizes and efficient, sublinear verification times. An attractive feature of zkSNARKs is their ability to generate a prover and verifier automatically from an *arithmetic circuit* that represents the computation, facilitating a broad spectrum of applications from privacy-preserving transactions on blockchain networks to secure authentication systems without revealing sensitive information.

2.4 Threat Model

In constructing a data availability layer for Bitcoin, it is imperative to consider a threat model that encompasses various adversarial capabilities aimed at compromising the system's integrity, availability, and security. Key threats include Sybil attacks, where adversaries could flood the network with malicious validators; data withholding attacks, where crucial transaction data is hidden to prevent or delay verification processes; and eclipse attacks, targeting specific nodes to isolate and feed them false information. Other significant concerns involve censorship, where powerful entities might selectively block or delay transactions, and Denial of Service (DoS) attacks, aimed at overwhelming the network's infrastructure, thereby degrading performance and accessibility. To safeguard against these threats, our design must incorporate robust security measures such as Sybil resistance, data redundancy, secure networking protocols, and mechanisms to ensure transaction inclusion, alongside active monitoring and responsive countermeasures to swiftly mitigate potential attacks.

3 OVERVIEW

This section briefly describes the overview of *Nubit DA*. As shown in Figure 1, at a high level, *Nubit DA* is composed of four critical components: namely, a set of validators, full storage nodes, and light clients. This section provides a summary of the functionality of each component. It delves into the life-cycle of user interaction within *Nubit DA*, offering a concise overview of the system.

- **Validators** These nodes operate using a consensus algorithm rooted in Practical Byzantine Fault Tolerance (Tendermint) and are tasked with proposing blocks and verifying the integrity of blocks and transactions. The intricacies of the consensus algorithm are detailed in Section 4.
- **Full Storage Node** After receiving block data from validators, these nodes are entrusted with the reliable storage of all data. The integrity and availability of stored data are critical, especially given the risks of malicious activities such as data withholding or tampering. To mitigate these risks, Data Availability Sampling (DAS) requests from light clients are employed to verify data availability, ensuring the system's resilience against such threats.
- **Light Client** Light clients obtain block headers broadcast by validators, which include data commitments. Based on these commitments, they may randomly initiate requests to full storage nodes to verify data availability. The process and significance of data availability sampling between full storage nodes and light clients are thoroughly examined in Section 4.3.

We further leverage a concrete use case to go over a complete system life-cycle depicted in Figure 1. Suppose Alice wants to complete a transaction (e.g., inscription, rollup data, etc.) using *Nubit DA*'s DA service. As outlined in step **1**, to submit a transaction, Alice must first dispatch her data and transaction metadata such as the address and nonce, to the validators for incorporation into the memory pool. Step **2** illustrates the process where validators, upon reaching a consensus, propose the block and its header. The block header includes the commitment to the data and its associated Reed-Solomon Coding (RS Code), whereas the block itself contains the original data, the corresponding RS Code, and basic transaction details. In the final phase, step **3**, the life-cycle concludes with data retrieval by Alice. Here, the light client downloads the block header, while the full storage node acquires both the block and its header. Light clients undertake the data availability sampling process to verify data availability. Moreover, after a threshold number of blocks has been proposed, a checkpoint of this history is recorded on the Bitcoin blockchain via Bitcoin anchoring. This ensures the validator set can thwart potential long-range attacks and enable fast unbonding.

In the subsequent sections, we will delve into the components of *Nubit DA* in greater detail.

4 CONSENSUS PROTOCOL

Nubit DA aims to fully inherit the security of Bitcoin, including economic security, immutability, and censorship resistance. It achieves this through the implementation of Bitcoin's native staking and anchoring methods, such as those introduced by Babylon. However, to reach a level of resistance to censorship comparable to Bitcoin's, a better consensus algorithm is needed to enable a larger validator set. *Nubit DA* explores an efficient Tendermint-based consensus powered by SNARK for signature aggregation. Because it will be inefficient for every node to download entire blocks to ensure data availability. So *Nubit DA* also integrates Data Availability Sampling (DAS) to scale the network with full storage nodes and light clients.

4.1 Tendermint-based Consensus

Overview. Byzantine Fault Tolerance (BFT) [1] is a consensus protocol engineered to achieve agreement among distributed nodes within a network. This ensures the maintenance of a uniform state across the system, even when faced with nodes either malfunctioning or acting maliciously. Distinct from blockchains' typically slow confirmation processes, BFT focuses on efficiency and finality. It allows a system to function correctly even if up to $\lfloor \frac{n-1}{3} \rfloor$ of its nodes are compromised, thereby safeguarding the system's integrity and operational continuity. Among the most recognized forms of BFT is *Tendermint*, which operates through a systematic series of communication stages: *propose*, *pre-vote*, *pre-commit*, and *commit*. This structured approach guarantees the accurate processing and logging of transactions across all non-faulty nodes. Nevertheless, the scalability of Tendermint-based consensus mechanisms diminishes as the number of nodes (i.e., validators) increases. This is attributed to the necessity for unanimous voting to reach a consensus,

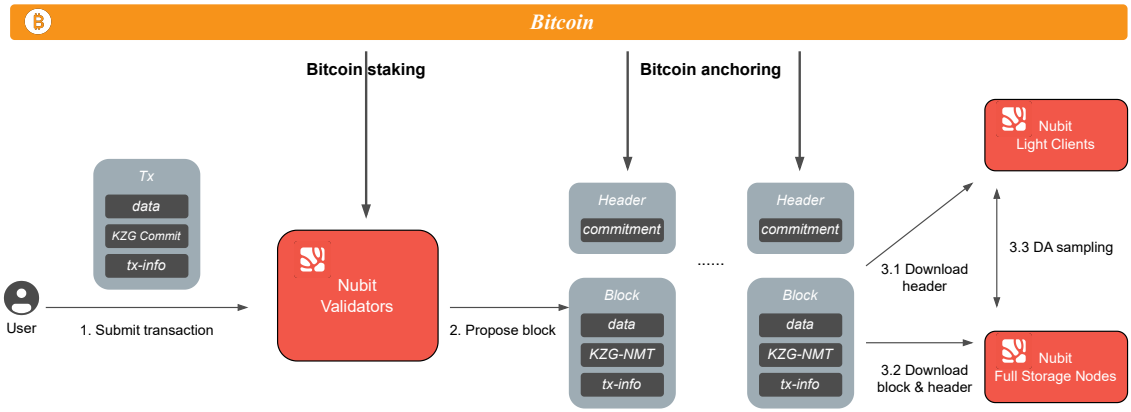

Figure 1: Overview of *Nubit DA*

requiring every node to gather votes from all other nodes to finalize a consensus. Consequently, the communication overhead for reaching a consensus decision escalates to $O(n^2)$ within the network. With a scenario of over 200,000 nodes, it becomes impractical for each node to collect votes from all others without significant enhancements to the consensus protocol.

NubitBFT from SNARK-based Signature Aggregation. Signature aggregation techniques merge multiple signatures into one, which is crucial for reducing communication and verification demands in systems with many validators. The BLS signature scheme is a well-known method that facilitates easy aggregation of signatures.

While combining signatures is straightforward in the BLS framework, identifying which validators have signed requires an additional mechanism, such as bitfields, which Ethereum employs. These bitfields serve as a checklist, indicating which validators have participated in signing. In this binary system, a '1' at a specific position suggests that the validator corresponding to that position has signed. These bitfields are always paired with their proofs, enabling a receiver to authenticate the proof and confirm that the bitfield represents a sequence of legitimate signatures and their combinations.

Although aggregating signatures is simple, merging two bitfields into one can be complex, and the verifier needs to be aware of the number of aggregations a signature has undergone. Recursive SNARKs offer a solution by allowing for the verification and aggregation of signatures through a straightforward approach. Each validator combines the SNARKs from neighboring validators and shares the updated SNARK, enabling rapid dissemination of signature shares across the network via the gossip protocol. This requires only $O(\log N)$ rounds of broadcasting for the voting process to be completed. Furthermore, using SNARK-based aggregation minimizes message sizes and communication costs.

Algorithm 1 shows the core outline of *Nubit DA*'s signature aggregation. As the process continues, validator V_i combines SNARK proofs and bitfields it receives and shares the updated versions with its neighbors:

- To begin, validator V_i creates a SNARK proof $\pi_i = \mathsf{PoK}\{\mathsf{pk}_i, \mathsf{H} : s = \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{H})\}$, where H is the block header, pk_i and sk_i is the public and private keys of validator V_i, respectively, to show that it has signed the block.
- It sends out this SNARK proof π_i along with a bitfield vec_i to all neighboring validators.

Meanwhile, validator V_i keeps aggregating SNARK proofs and bitfields and then broadcasts them out to all neighbors:

- When validator V_i gets (π_j, vec_j) from a neighboring validator V_j, it first checks if the proof π_j is valid. If validated, it merges its current proof and bitfield with those received, resulting in an updated proof $\pi_i' = \mathsf{Aggregate}(\pi_i, \mathsf{vec}_i, \pi_j, \mathsf{vec}_j)$ and an updated bitfield $\mathsf{vec}_i' = \mathsf{vec}_i \mid \mathsf{vec}_j$.
- Periodically, validator V_i broadcasts the most recent combination of (π_i, vec_i) to its neighbors.
- This process of aggregation halts when the sum of the entries in vec_i reaches or exceeds two-thirds of the total number of validators, i.e. $\mathsf{Sum}(\mathsf{vec}_i) \geq 2/3N$. At this point, the SNARK proof π_i can confirm that the block has been signed by at least k validators such that $k \geq 2/3N$.

Algorithm 1 SNARK-Based Signature Aggregation

procedure SIGN(V_i, r)
  Input: Validator V_i, Broadcast Interval r
  Output: $\top$ until the block has been signed
  $\mathsf{vec} \leftarrow (\mathsf{vec}_k)_{|\mathsf{vec}_k=1 \text{ if } k=i \text{ otherwise } \mathsf{vec}_k=0}$ ▷ Initialize bitfield
  $\pi_i \leftarrow \mathsf{PoK}\{\mathsf{pk}_i, \mathsf{H} : s = \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{H})\}$ ▷ Create a SNARK proof π_i
  $\mathsf{broadcast}(\pi_i, \mathsf{vec}_i)$ ▷ Broadcast initially
  while $\Sigma_{i \in N} \mathsf{vec}_i \leq 2N/3$ **do**
    $(\pi_j, \mathsf{vec}_j) \leftarrow \mathsf{receive}()$ ▷ Receive update
    if $\mathsf{valid}(\pi_j)$ **then**
      $\pi_i \leftarrow \mathsf{aggregate}(\pi_i, \mathsf{vec}_i, \pi_j, \mathsf{vec}_j)$ ▷ Update proof
      $\mathsf{vec}_i \leftarrow \mathsf{vec}_i \mid \mathsf{vec}_j$ ▷ Update bitfield
    if $\mathsf{timer.elapsed}() \geq r$ **then**
      $\mathsf{timer.reset}()$
      $\mathsf{broadcast}(\pi_i, \mathsf{vec}_i)$ ▷ Broadcast periodically
  return $\top$

4.2 Block Assembly

Overview. To conclude that the overall data of the block is available by sampling only part of the information of a block (i.e. chunks), we need to use erasure code to encode the block. To ensure the correctness of the encoding, we need KZG commitment to be generated and included in the block header. The entire block can be large and introduce extreme communication costs. To mitigate this issue, block dispersion is applied on each block.

Block Dispersion. The process starts with the leader distributing the block data to ensure every network node has access to it. Having every validator hold a complete copy of the block would be too costly. So, the leader broadcasts the coded chunks, as described before. These coded chunks are then shared among different groups of validators. Each group is responsible for keeping and managing specific chunks, which they do by subscribing to particular topics. To ensure that there are always enough active validators in each group to keep the communication stable and reliable, systems like reputation scores and slashing penalties are put in place.

Erasure Code (EC) and Data Attestation. Erasure coding is a strategy that introduces extra data to safeguard against the loss of original data, crucial for storing and sending data. A common type of erasure coding used is Reed-Solomon (RS) coding. With RS coding, each network participant needs to store only a small section of the block, called a chunk, while ensuring the data's integrity. Essentially, the lead node divides the block's data into a grid of $k \times k$ chunks. This grid is then enlarged through 2D-RS coding to form a $2k \times 2k$ grid of chunks. Following this, the leader creates a KZG commitment for each chunk's data and proofs. This setup prevents unauthorized modification of the chunks in transit, as the commitment makes it impossible to generate valid proof for any altered chunk.

Note that the KZG commitment is included in the block header, also known as Data Attestation. This allows for the direct verification of chunks, leaving no possibility for incorrect encoding.

4.3 Data Availability Sampling

Overview. Since *Nubit DA* enables light clients as part of its consensus protocol, it is crucial to ensure the data integrity held by those light clients. Data Availability Sampling (DAS) is a technique that enables a participant to verify the presence of block data without needing to download the entire block. This is done by conducting multiple rounds of random sampling on small portions of the block data. Each successful sampling round increases the likelihood that the data is fully available. Once a predetermined confidence level is reached, the block data is deemed accessible. Validators, full storage nodes, and light nodes each have specific roles in ensuring data availability through their designated protocols.

Protocols. There are two main protocols involved in DAS:

- The *sampling protocol* is executed between a verifier and the data source (the validators or a full storage node). Starting with the KZG commitment from the block header, the verifier asks for a sufficient quantity of block chunks selected randomly from the source. If all requested chunks are received and match the KZG commitment, the verifier concludes the check as successful.
- The *decoding protocol* is carried out by a decoder working with the validator group. It also begins with the KZG commitment and involves requesting block chunks from the validators. When it gathers more than a specific percentage of the total chunks, all verified to be correct, the decoder reconstructs the full block through RS decoding.

Participants. Three types of parties participate in DAS:

- Validator: Validators are responsible for running the *sampling protocol* within the validator set. A validator will sign the block header only if the output of the *sampling protocol* is successful, thereby ensuring the data's availability within the validator group.
- Full Storage Node: Once a block is finalized, full storage nodes take on the entire block by employing the *decoding protocol* with the validator set. These nodes then respond to chunk requests from light clients.
- Light Client: Light clients obtain the block header from a validator and engage in the *sampling protocol* with a full storage node. If the protocol is successful after sufficient sampling, the block is considered available, and the full storage node's reputation is enhanced.

4.4 Bitcoin Inheritance

Overview. Nubit DA aims to fully inherit the security of Bitcoin, including economic security and immutability based on PoW. Achieving this integration involves leveraging Bitcoin staking and anchoring within the protocol framework. *Nubit DA*'s efficient consensus protocol makes it possible to achieve a level of resistance to censorship that matches that of Bitcoin.

Bitcoin Staking. The Bitcoin staking approach enables Bitcoin owners to participate in Proof of Stake (PoS) blockchains directly, bypassing the need for third-party services for custody, bridges, or token wrapping. This method offers strong economic security measures that are enforceable within PoS networks while allowing for the quick release of staked assets to improve liquidity for those staking their Bitcoin.

Using the Babylon Bitcoin staking, *Nubit DA* incorporates extractable one-time signatures (EOTS). This technology ensures accountability, with the premise that duplicating signatures for different blocks at the same level leads to the disclosure of the secret key. In conditions where stakers act in good faith, they receive earnings from block rewards and transaction fees.

Bitcoin anchoring. For PoS networks, the period required to unbond staked assets is typically extended to guard against long-range attacks, which entail minimal costs for attackers who wish to create alternative chain forks post-unbonding. To mitigate these attacks while facilitating fast unbonding, *Nubit DA* has checkpoints in its blockchain. These checkpoints invalidate any forks that originate before them. This security measure, known as Bitcoin anchoring and pioneered by Babylon, records both block hashes and the votes of the staking set on the Bitcoin blockchain.

Through Bitcoin anchoring, *Nubit DA* drastically reduces the withdrawal timeframe to less than four hours from weeks. Additionally, these checkpoints provide an extra layer of security guarantee, such that the integrity of data stored in a full storage node can be

determined based on these checkpoints. Even in the event of a complete *Nubit DA* collapse, nodes can still perform data restoration using full nodes and checkpoints submitted on Bitcoin.

5 RELATED WORK

In this section, we will introduce the related work on the main components of the system.

Consensus Protocol. The consensus problem [19] was first proposed by Shostak, Pease, and Lamport as a fundamental issue in distributed computing. It has attracted widespread attention, and many different variants of the consensus problem have been studied, resulting in notable research such as Paxos [18], RAFT [22], PBFT [11], and many others [7, 14, 15]. Among these, compared to Paxos and Raft, which can only tolerate crash faults, BFT protocols, starting with PBFT, are capable of handling any corrupt nodes, even if they are malicious. Based on PBFT, many subsequent protocols have emerged. Some protocols employ *optimistic execution* to enhance network performance, such as [1, 5, 17, 25], requiring the network to be fault-free and clients to have minimal disputes. Conversely, some protocols [4, 6, 8, 12, 13, 24] emphasize maintaining satisfactory performance even when the system is under attack or suffers severe faults—at the cost of performance. Among these, HoneyBadgerBFT [20] is considered the first practical asynchronous BFT protocol that guarantees network liveness without any timing assumptions. Tendermint [10] is the BFT most favored in the web3 community, modernizing existing work and simplifying the BFT algorithm via a peer-to-peer gossip protocol. Building on these studies, consensus algorithms based on Tendermint have become one of the foundational algorithms for blockchains, as blockchains require consensus protocols to tolerate a significant number of malicious nodes while maintaining rapid final confirmation—strengths of the Terdermint algorithm. Compared to the aforementioned studies, our research proposes a solution to optimize the actual communication complexity of Tendermint in the blockchain context: although nodes still need to communicate with all other nodes, the volume of communication required to reach consensus can be significantly reduced.

Data Availability. Blockchain data availability was first discussed by [3], with subsequent work [2] further discussing how a separate data availability layer could be deployed as part of the blockchain network. Moreover, [27] introduced a method to recover data that may be tampered with by malicious nodes using a Coded Merkle Tree, which utilizes a series of sparse matrices to construct erasure codes. [23] proposed the possibility of optimizing the volume of block broadcast communication, where each storage node only needs to receive 1/N of the data to ensure the overall network's data availability. [21] discussed the relationship between data availability and blockchain second-layer aggregated commitments. The aforementioned works all discuss DAS (Data Availability Schemes) on an informal level, while [16] is the first to formally define DAS technology. Our work adopts a data commitment scheme based on KZG Commitments, which, compared to schemes based on Merkle Trees and their variants, generates commitments of constant size, significantly reducing the block header size that light nodes need to download.

6 CONCLUSION

In conclusion, this paper introduces *Nubit DA*, a groundbreaking advancement in the Bitcoin ecosystem, addressing critical challenges that have long hampered its scalability and efficiency. By ingeniously leveraging a signature scheme based on Zero-Knowledge Proofs (ZKP), *Nubit DA* significantly reduces the communication overhead among validators, enabling the network to achieve instant finality. This innovative approach not only enhances the scalability of the Bitcoin network by supporting a larger volume of transactions but also maintains the core principles of decentralization and security that Bitcoin is known for.

REFERENCES

[1] Michael Abd-El-Malek, Gregory R Ganger, Garth R Goodson, Michael K Reiter, and Jay J Wylie. 2005. Fault-scalable Byzantine fault-tolerant services. *ACM SIGOPS Operating Systems Review* 39, 5 (2005), 59–74.

[2] Mustafa Al-Bassam. 2019. LazyLedger: A Distributed Data Availability Ledger With Client-Side Smart Contracts. *CoRR* abs/1905.09274 (2019). arXiv:1905.09274 http://arxiv.org/abs/1905.09274

[3] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. 2019. Fraud and Data Availability Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities. arXiv:1809.09044 [cs.CR]

[4] Yair Amir, Brian Coan, Jonathan Kirsch, and John Lane. 2010. Prime: Byzantine replication under attack. *IEEE transactions on dependable and secure computing* 8, 4 (2010), 564–577.

[5] Yair Amir, Claudiu Danilov, Danny Dolev, Jonathan Kirsch, John Lane, Cristina Nita-Rotaru, Josh Olsen, and David Zage. 2008. Steward: Scaling byzantine fault-tolerant replication to wide area networks. *IEEE Transactions on Dependable and Secure Computing* 7, 1 (2008), 80–93.

[6] Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Quéma. 2013. Rbft: Redundant byzantine fault tolerance. In *2013 IEEE 33rd international conference on distributed computing systems*. IEEE, 297–306.

[7] Catalonia-Spain Barcelona. 2008. Mencius: building efficient replicated state machines for WANs. In *8th USENIX Symposium on Operating Systems Design and Implementation (OSDI 08)*.

[8] Alysson Bessani, João Sousa, and Eduardo EP Alchieri. 2014. State machine replication for the masses with BFT-SMART. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 355–362.

[9] Nakamoto S Bitcoin. 2008. Bitcoin: A peer-to-peer electronic cash system.

[10] Ethan Buchman, Jae Kwon, and Zarko Milosevic. 2019. The latest gossip on BFT consensus. arXiv:1807.04938 [cs.DC]

[11] Miguel Castro, Barbara Liskov, et al. 1999. Practical byzantine fault tolerance. In *OsDI*, Vol. 99. 173–186.

[12] Allen Clement, Manos Kapritsos, Sangmin Lee, Yang Wang, Lorenzo Alvisi, Mike Dahlin, and Taylor Riche. 2009. Upright cluster services. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. 277–290.

[13] Allen Clement, Edmund Wong, Lorenzo Alvisi, Mike Dahlin, Mirco Marchetti, et al. 2009. Making Byzantine fault tolerant systems tolerate Byzantine faults. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*. The USENIX Association.

[14] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)* 32, 2 (1985), 374–382.

[15] Juan Garay and Aggelos Kiayias. 2020. Sok: A consensus taxonomy in the blockchain era. In *Topics in Cryptology–CT-RSA 2020: The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24–28, 2020, Proceedings*. Springer, 284–318.

[16] Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. 2023. Foundations of Data Availability Sampling. Cryptology ePrint Archive, Paper 2023/1079. https://eprint.iacr.org/2023/1079 https://eprint.iacr.org/2023/1079.

[17] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. 2007. Zyzzyva: speculative byzantine fault tolerance. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*. 45–58.

[18] Leslie Lamport. 2019. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*. 277–317.

[19] Leslie Lamport, Robert Shostak, and Marshall Pease. 2019. The Byzantine generals problem. In *Concurrency: the works of leslie lamport*. 203–226.

[20] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. 2016. The honey badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 31–42.

[21] Kamilla Nazirkhanova, Joachim Neu, and David Tse. 2021. Information Dispersal with Provable Retrievability for Rollups. Cryptology ePrint Archive, Paper 2021/1544. https://eprint.iacr.org/2021/1544 https://eprint.iacr.org/2021/1544.

[22] Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In *2014 USENIX annual technical conference (USENIX ATC 14)*. 305–319.
[23] Peiyao Sheng, Bowen Xue, Sreeram Kannan, and Pramod Viswanath. 2021. ACeD: Scalable Data Availability Oracle. arXiv:2011.00102 [cs.CR]
[24] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. 2009. Spin one's wheels? Byzantine fault tolerance with a spinning primary. In *2009 28th IEEE International Symposium on Reliable Distributed Systems*. IEEE, 135–144.
[25] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. 2010. EBAWA: Efficient Byzantine agreement for wide-area networks. In *2010 IEEE 12th International Symposium on High Assurance Systems Engineering*. IEEE, 10–19.
[26] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.
[27] Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. 2019. Coded Merkle Tree: Solving Data Availability Attacks in Blockchains. arXiv:1910.01247 [cs.CR]

A SUCCINCT NON-INTERACTIVE ARGUMENTS OF KNOWLEDGE (SNARK).

A SNARK scheme $\mathsf{SNARK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vrf})$ includes the following protocols.

- $\mathsf{crs} \leftarrow \mathsf{SNARK.Setup}(C)$ takes as input C, which is a circuit depicting a relation $\mathcal{R}_C$ that $(\phi, w) \in \mathcal{R}_C$ if and only if $C(\phi, w) = 1$. It outputs a common reference string crs.
- $\pi \leftarrow \mathsf{SNARK.Prove}(\mathsf{crs}, \phi, w)$ takes as input crs, a public input (a.k.a. , instance), and a private input (a.k.a., witness) such that $C(\phi, w) = 1$. It returns a proof π.
- $b \leftarrow \mathsf{SNARK.Vrf}(\mathsf{crs}, \phi, \pi)$ takes as input crs, a public input ϕ, and a proof π. It returns a bit $b \in \{0, 1\}$ such that $b = 1$ if and only if the prover (i.e., the generator of π) knows w that $C(\phi, w) = 1$.

As SNARK, the proof size and verification time should be polylogarithmic in the circuit size for SNARKs. Notably, for some SNARKs, including our decided SNARK, the two are constant. To omit the descriptions for setups, an argument (or proof) of knowledge is written as $\mathsf{PoK}\{w : C(\phi, w) = 1\}$.

B KATE-ZAVERUCHA-GOLDBERG (KZG) COMMITMENT

As one of the building blocks of SNARKs, the KZG commitment, named after its inventors Kate, Zaverucha, and Goldberg, is a form of cryptographic commitment scheme based on elliptic pairings and polynomial commitments. A commitment scheme allows one to commit to a chosen value while keeping it hidden from others, with the ability to reveal the committed value later. KZG commitments specifically enable efficient and secure commitments to polynomials, allowing one to commit to a polynomial without disclosing it, and later prove properties about the polynomial (such as its value at a certain point) without revealing the entire polynomial. KZG commitment has a few stages to perform its functionality:

- **Setup** A trusted setup involves the selection of a secret scalar s and the generation of public parameters $[s^d]_1$ in group G_1 and $[s^d]_2$ in group G_2, corresponding to a maximum polynomial degree d. Here, $[s]_1$ is defined by the operation sG, where G is a generator of G_1, and $[s]_2$ is defined by sH, where H is a generator of G_2. These groups, G_1 and G_2, are interconnected through a elliptic pairing function e, formalized as $e : G_1 \times G_2 \rightarrow G_T$, where G_T denotes the target group for the pairing. Following the completion of the setup process, the secret s is securely discarded to ensure the integrity of the system.
- **Commitment** The prover selects a polynomial $P(x)$ of degree d, commits to it via: $C(P) = [P(s)]_1$, using $[s^d]_1$.
- **Proof Generation** The prover wants to prove $P(a) = b$ for some a and b, compute a witness polynomial $W(x)$ such that $P(x) - b = (x - a)W(x)$. The proof π is the commitment to $W(x)$, as $\pi = [W(s)]_1$.
- **Verification** To verify b as the value of P at a, check if the elliptic pairing equation holds:

$$e(\pi,\ [s - a]_2) = e(C(P) - [b]_1,\ H)$$

After the verification stage, the verifier could accept or reject the proof provided by the prover. The KZG commitment brings many benefits to be applied to blockchain communication:

- **Succinctness** The size of commitments and proofs does not grow with the size of the polynomial $P(x)$, enhancing scalability.
- **Hiding** The commitment $C(P)$ does not reveal any information about $P(x)$ other than the committed value at a specific point, assuming the hardness of the discrete logarithm problem in G.
- **Binding** Assuming it is hard to find s, it is computationally infeasible to find a different polynomial $P'(x)$ that would result in the same commitment for a different evaluation point.