Stateless Blockchain

Submitted in partial fulfillment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

With specialization in

CYBER LAW AND INFORMATION SECURITY

Submitted by

Vadadoriya Nitinbhai Gokalbhai
(Enrollment No. MCL2023003)

Under the Supervision of

Dr.S.Venkatesan Sir

to the

DEPARTMENT OF INFORMATION TECHNOLOGY

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
ALLAHABAD

December, 2024

Certificate

It is certified that the work contained in the thesis titled “Stateless Blockchain” by “Vadadoriya Nitinbhai Gokalbhai” has been carried out under my supervision and that this work has not been submitted else where for a degree.

Dr.S.Venkatesan Sir
Department of Information Technology
IIIT Allahabad

CANDIDATE DECLARATION

I, Vadadoriya Nitin, MCL2023003, certify that this thesis work titled "Stateless Blockchain" is submitted by me towards partial fulfillment of the requirement of the Degree of Master of Technology in the Department of Information Technology, Indian Institute of Information Technology, Allahabad.

I understand that plagiarism includes:

- Reproducing someone else's work (fully or partially) or ideas and claiming it as one's own.
- Reproducing someone else's work (verbatim copying or paraphrasing) without crediting.
- Committing literary theft (copying some unique literary construct).

I have given due credit to the original authors/sources through proper citation for all the words, ideas, diagrams, graphics, computer programs, experiments, results, and websites that are not my original contributions. I have used quotation marks to identify verbatim sentences and given due credit to the original authors/sources.
I affirm that no portion of my work is plagiarized. In the event of a complaint of plagiarism, I shall be fully responsible.

Vadadoriya Nitin
MCL2023003
Department of Information Technology
IIIT Allahabad
Prayagraj - 211015, U.P.
Date: 05/12/2024

Acknowledgements

I reserve a special place in my heart for my beloved parents, whose unwavering love, unwavering support, and unwavering belief in my abilities have been the bedrock upon which my dreams have flourished. Their persistent support, sacrifices, and unshakable trust in my abilities have been the driving factors behind my quest for knowledge and academic pursuits.

I owe a tremendous debt of gratitude to my esteemed supervisor, **Dr.S.Venkatesan Sir**, whose guidance and advice have been the compass guiding me through the many twists and turns of this thesis. His stimulating conversations, insightful feedback, kind advice, and boundless forbearance have challenged me to push the boundaries of my capabilities and inspired me to strive for academic excellence. I am very thankful for the trust you put in me and the chances you gave me to grow both professionally and personally. I am grateful beyond words for the opportunity to have worked under your guidance, and I hope my thesis serves as a fitting tribute to your hard work, knowledge, and encouragement.

Finally, I want to thank the Faculty who helped me grow as a scholar.

Vadadoriya Nitin

ABSTRACT

We propose a solution for stateless blockchain that significantly reduces storage cost on the miner side as well as computation cost on the user side. Our solution is for a UTXO based blockchain like bitcoin, where every miner needs to maintain UTXO state for transaction validation and execution. In our solution we are using a Hash tree accumulator that is based on the merkle tree concept that is used for transaction validation. To mitigate double spend attack we are using a polynomial hash accumulator that accumulates all input/spent transactions and using the latest accumulator we can check if the input transaction is part of the latest accumulator or not. Miners only need to store the latest accumulator and chain of transactions commitment instead of the whole state to mine blocks on blockchain.

Contents

Acknowledgements **iv**

Abstract **v**

Contents **vi**

List of Figures **vii**

List of Tables **viii**

1 Introduction **1**
1.1 What is stateless? 2
1.2 How state can removed? 2

2 Literature Review **3**
2.1 Literature Table 4

3 WorkFlow **5**
3.1 System WorkFlow 6
3.1.1 User 6
3.1.2 Miner 6
3.1.3 Working 6

4 Result **8**
4.1 Hash Tree-Based Commitment 9
4.1.1 Time Complexity 9
4.1.2 Space Complexity 9
4.2 Polynomial Hash Accumulator 9
4.2.1 Representation of the Accumulator 9
4.2.2 Adding a New Element 10
4.2.3 Membership Verification 10
4.2.4 Time Complexity 10
4.2.5 Space Complexity 10

5 Conclusions and Future Scope **12**
5.1 Conclusions 13
5.2 Future Scope 13

References **14**

List of Figures

3.1 Block mining procedure 6
3.2 Example 7

List of Tables

4.1 Comparison of Blockchain Storage Requirements 11

Chapter 1

Introduction

Blockchain is an immutable distributed ledger that replicates and shares data between peer to peer network.As of September 2024 the bitcoin blockchain size is approximately 552 GB, while the ethereum blockchain is around 1160 GB when fully synced using default settings. Both chains continue to grow as more blocks are added to their respective networks [1].The validation state size refers to the data that a node needs to validate new transactions and blocks on the blockchain. The validation state for bitcoin, known as the UTXO(Unspent Transaction Output) set, is significantly smaller than the full blockchain size. As of 2024, the UTXO set is estimated to be around 4 to 5 GB [4]. because of this large storage size the fewer people will run full nodes. For that reason stateless blockchain research comes into the picture. A stateless blockchain is a design that removes the requirement for nodes to store the entire blockchain state. Instead, nodes rely on cryptographic proofs, known as state witnesses, which provide the necessary data to verify transactions. To implement it we have several concepts like Merkle Trees, RSA based accumulator etc. but these all solution have some problem related to witness. user need to continuously update their witness, witness modification, user storage burden etc. so we propose solution that is used witness free accumulator to mitigate all above problem. also we are using Hash tree commitment based on merkle tree concept for mitigate continuously witness update problem on user side.

1.1 What is stateless?

Stateless mean a network that validate and execute transactions without holding state/data related to users,account and balances. there might be required some storage for store necessary information related to network and validate transactions but it should be very less compare tho whole state.

1.2 How state can removed?

Using cryptography approach we can accumulate whole state in constant size that later on use for validate and execute transaction. for successfully validate transaction, user need to additionally provide proof for their transactions. popular accumulator is RSA based accumulator, pairing-based accumulator, Merkle tree based.

Chapter 2

Literature Review

2.1 Literature Table

Authors	Paper	Findings
SHRAVAN YUPENG ALEXANDER CHARALAMPOS	Edrax : A Cryptocurrency With Stateless Transaction Validation (2018)	This paper present new distributed vector commitment construction used in the implementation of account-based EDRAX and sparse Merkle tree representing UTXO based models. in both approach user need to continuously monitor network and based on every transaction their proof need to update.
MIRANDA CHRIST JOSEPH BONNEAU .	Limits on revocable proof systems, with applications to stateless blockchains (2020)	Understand trade-off between reduce storage cost and increase burden on users.
B SWAROOPA REDDY, T UDAY KIRAN REDDY.	CompactChain: an efficient stateless chain for UTXO-model blockchain (2024)	Proposed solution for UTOX based stateless blockchain using two commitment, one for spent transactions and other for unspent transactions. efficiently verify and add new transaction in commitment with in constant time. in this solution user need to update their proof for every transaction executions.
HAOJUN XINBO HONGRU XUBO.	Merkle Tree: A Fundamental Component of Blockchains (2021)	The paper proposes that Merkle trees are essential for ensuring secure and efficient transaction verification in blockchain systems, like Bitcoin and Ethereum, by using cryptographic hashing.
GUNTER SAAKE MUHAMMAD SAQIB.	Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data (2015)	In this paper they used merger concept of Merkle tree and B+ tree to ensure integrity of outsourced data. they have not write about how deletion of data work.

Chapter 3

WorkFlow

3.1 System WorkFlow

.

3.1.1 User

user wallet will store all UTXO along with respective proof. this proof for to convince miner to this UTXO is part of UTXO state. When were new UTXO get generated it will sent to respective user wallet along with respective proof. miner is responsible for do this.

3.1.2 Miner

Miner have complete chain of only header. latest block contain latest commitment of all transactions which are either spent or unspent. this commitment help to validate transactions. if Alice claiming that i have 1 BTC input transaction, miner can check this thing is true or false using this commitment. block header also have input transactions accumulator that help to prevent again double spending attack. when were any input transaction got successfully spent then it will be added in this accumulator so, later if miner get again this transaction as input, miner can failed it execution using this accumulator.

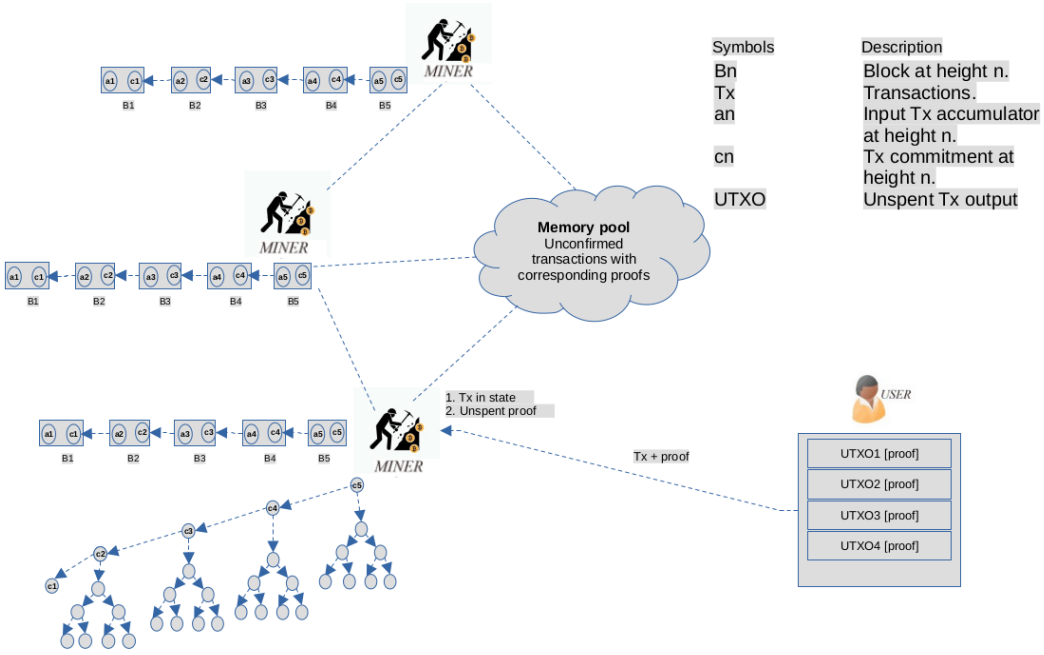

Figure 3.1: Block mining procedure

3.1.3 Working

let assume Alice want to spent transaction Tx5, process will be done in following step.

1. Alice send Tx5 with proof to miner.
2. miner collect transactions, let say miner take Tx5 for execution.

2.1 miner first validate Tx5 by recalculate commitment using provided proof and check this commitment c3 exit in miner blockchain or not. Tx5 is in state so it successful passed.

2.2 miner need to ensure this input Tx5 is already spent or not, for that miner used Polynomial Hash Accumulator to check. since Tx5 is not spent yet so miner execute this transaction, let say four new output transaction generated [tx9,tx10,tx11,tx12]. so miner will add these all new transactions in commitment using merkle tree (right now we are generating merkle tree commitment using four transactions). miner also responsible for generating proof for every new generated output transaction. let say Tx10 is remaining coin that will send to Alice return with proof like [H9,H105,c3]. miner also add spend transaction Tx5 in accumulator.

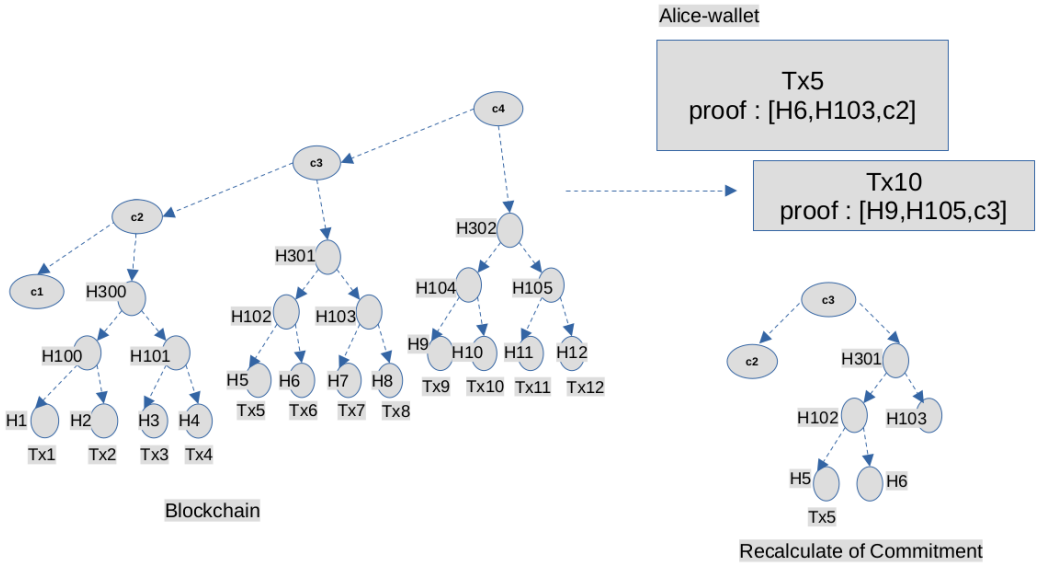

Figure 3.2: Example

- c1,c2,c3 and c4 all are commitment that are store in block.
- Block contain Hash of previous block, commitment, accumulator, version, Block difficulty, timestamp, nonce.
- mine block will be broadcast to network and other miner validate block's transactions for accept it.

Chapter 4

Result

4.1 Hash Tree-Based Commitment

Hash tree-based commitments utilize the concept of Merkle trees to generate a commitment. However, they differ from standard Merkle trees in structure and functionality.

In a standard Merkle tree, which is a full binary tree, calculating the root commitment requires an equal number of hashes on both the left and right sides at every level. In contrast, a hash tree used for commitment has a fixed structure on the right side and dynamic structure on left side. For instance, the right side of the tree contains 2^l hashes, where l represents the height of the merkle tree. In our example, l = 2, but it can be increased depending on the use case.

Increasing l comes with a trade-off: while it enhances throughput, it also increases the proof size for the user. On the right side of the hash tree, the number of hashes can vary and does not need to match the left side.

This design provides flexibility but requires careful consideration of the trade-offs between efficiency, proof size, and throughput.

4.1.1 Time Complexity

To generate the commitment of 2^l output transactions, it requires O(M * log(M)) time, where M = 2^l.

Similarly, using sibling nodes, we can efficiently recalculate the commitment using exactly l+1 hashes in O(log(M)) time. Therefore, if we have P input transactions, we need O(P * log(M)) time to validate them.

Here, we assume that O(1) time is required to check whether the recalculated commitment exists in our database in the average case using a hash table.

4.1.2 Space Complexity

In our proposed solution, the block header size is 112 bytes. An accumulator of 32 bytes is additional compared to the Bitcoin block header.

Since miners only store the block headers, they require O(112 * H * bytes) of disk space, where H is the height of the longest chain.

4.2 Polynomial Hash Accumulator

A Polynomial Hash Accumulator represents a set of elements using a polynomial whose roots correspond to the elements of the set. It allows efficient verification of membership and supports the addition of new elements. Here's a detailed breakdown of how it works, how new elements are added, and how to check for membership.

4.2.1 Representation of the Accumulator

The accumulator is represented as a polynomial:

$$P(x) = (x-a)(x-b)(x-c)\ldots \mod N$$

where $a, b, c, \ldots$ are the elements to be accumulated, and N is a prime or an RSA modulus.

The accumulator's state is stored as the coefficients of $P(x)$, or, in certain cryptographic contexts, as a single evaluation $P(g)$, where g is a group generator.

4.2.2 Adding a New Element

When a new element d is added:

Update the Polynomial

$$P_{\text{new}}(x) = P_{\text{old}}(x) \cdot (x - d) \mod N$$

This means the new polynomial $P_{\text{new}}(x)$ includes d as an additional root.

4.2.3 Membership Verification

To check if an element d is in the accumulator:

Compute the Polynomial Evaluation

Verify that $P(d) \equiv 0 \mod N$. If $P(x)$ is stored as a polynomial, this involves evaluating $P(x)$ at d:

$$P(d) = (d - a)(d - b)(d - c)\ldots \mod N$$

If $P(d) = 0$, then d is a root, and it belongs to the set.

4.2.4 Time Complexity

Adding a New Element

To add a new element d, you update the polynomial as:

$$P_{\text{new}}(x) = P_{\text{old}}(x) \cdot (x - d) \mod N$$

This involves multiplying the existing polynomial $P_{\text{old}}(x)$ (of degree n) by $(x - d)$, which is a linear operation. The time complexity for this multiplication is $O(n)$, where n is the degree of $P_{\text{old}}(x)$.

Membership Verification

To verify membership for an element d, you compute:

$$P(d) = (d - a)(d - b)(d - c)\ldots \mod N$$

If the polynomial $P(x)$ is stored explicitly as coefficients, this involves substituting d into $P(x)$ and evaluating it. Using Horner's method for polynomial evaluation, this can be done in $O(n)$, where n is the degree of the polynomial.

4.2.5 Space Complexity

Storing the Polynomia

If the polynomial $P(x)$ is stored explicitly, it requires $O(n)$ space, where n is the number of accumulated elements (degree of the polynomial).

Content	Stateful Blockchain	Edrax-Stateless Blockchain	Our-Stateless Blockchain
Complete BC	606.643 GB	2123.25 GB (since they are storing proof in block)	606.643 GB + (Number of blocks * 32 Byte)
AVG. Block header size	80 Byte	112 Byte	112 Byte
Miner UTXO state size	10.1 GB (approx.) Or 155 Million UTXO	Constant size commitment (32 Byte)	112 Byte * Number of blocks (0.97 GB) + spent tx's
Client state	Unspent txs.	Unspent txs + Number of unspent tx * Local Proof size (32 Byte) * 40	Unspent txs + Number of unspent tx * Local Proof size (32 Byte) * 3

Table 4.1: Comparison of Blockchain Storage Requirements

Chapter 5

Conclusions and Future Scope

5.1 Conclusions

After analyzing various solutions for stateless blockchains, I conclude that there is an inherent trade-off between the state burden on miners and the computational burden on users. More specifically, these solutions tend to reduce the state burden for miners while increasing either the state or computational burden on users. This trade-off arises because these solutions rely on witness-based accumulators.

One key challenge is that users can modify witnesses, making it difficult to ensure their correctness. This issue can be mitigated by using witness-free accumulators, such as polynomial hash-based accumulators, which eliminate the need for witnesses and address this problem effectively.

5.2 Future Scope

Using a hash tree, we can significantly reduce storage requirements on the miner's side while eliminating the need for users to update the proof of UTXO, making this solution practical to implement. However, for input transactions, we rely on a witness-free accumulator, which is currently expensive in terms of both time and space. To address this, we should either work on reducing its operational costs or explore alternative approaches to achieve the same functionality more efficiently.

References

[1] M. Ye, J. Zhang, S. Zhao, J. Liu, T. Liu, B. Du, and D. Tao, "Deepsolo: Let transformer decoder with explicit points solo for text spotting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 348–19 357.

[2] W. Feng, W. He, F. Yin, X.-Y. Zhang, and C. Liu, "Textdragon: An end-to-end framework for arbitrary shaped text spotting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[3] L. Qiao, S. Tang, Z. Cheng, Y. Xu, Y. Niu, S. Pu, and F. Wu, "Text perceptron: Towards end-to-end arbitrary-shaped text spotting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 11 899–11 907.

[4] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "Abcnet: Real-time scene text spotting with adaptive bezier-curve network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9809–9818.

[5] Y. Liu, C. Shen, L. Jin, T. He, P. Chen, C. Liu, and H. Chen, "Abcnet v2: Adaptive bezier-curve network for real-time end-to-end text spotting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, pp. 1–1, 2021.

[6] L. Qiao, Y. Chen, Z. Cheng, Y. Xu, Y. Niu, S. Pu, and F. Wu, "Mango: A mask attention guided one-stage scene text spotter," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 2467–2476.

[7] M. Huang, Y. Liu, Z. Peng, C. Liu, D. Lin, S. Zhu, N. Yuan, K. Ding, and L. Jin, "Swintextspotter: Scene text spotting via better synergy between text detection and text recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 4593–4603.

[8] X. Zhang, Y. Su, S. Tripathi, and Z. Tu, "Text spotting transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 9519–9528.

[9] C. K. Chng, Y. Liu, Y. Sun, C. C. Ng, C. Luo, Z. Ni, C. Fang *et al.*, "Icdar2019 robust reading challenge on arbitrary-shaped text-rrc-art," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 1571–1576.

[10] J. Weinman, Z. Chen, B. Gafford, N. Gifford, A. Lamsal, and L. Niehus-Staab, "Deep neural networks for text detection and recognition in historical maps," in *Proceedings of the IEEE International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 902–909.