

Obscuring or deleting data from nodes in a Blockchain network

Report of IT499 Major Project-II

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

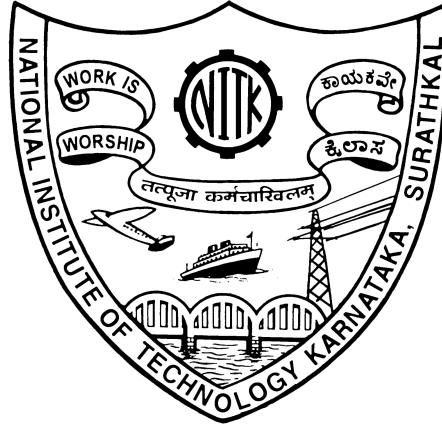
by

Nirmal Harshal Khedkar 181IT230

Jeeukrishnan Kayshyap 181IT220

under the guidance of

Dr. Ananthanarayana V. S.



DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025

April, 2022

DECLARATION

We hereby *declare* that the Project Work Report entitled “Obscuring or deleting data from nodes in a Blockchain network”, which is being submitted to the **National Institute of Technology Karnataka, Surathkal**, in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology in the Department of Information Technology, is a *bonafide report of the work carried out by us*. The material contained in this Report has not been submitted at any University or Institution for the award of any degree.

Registration Number, Name & Signature of the Student(s)

(1) 181IT230 - Nirmal Khedkar - 

(2) 181IT220 - Jeeukrishnan Kayshyap - 

Department of Information Technology


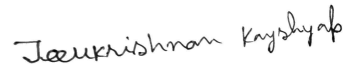
Place: NITK, Surathkal

Date: 2/04/2022

CERTIFICATE

This is to *certify* that the Project Work Report entitled “Obscuring or deleting data from nodes in a Blockchain network”, submitted by

Sl. No., Registration Number & Name of the Student(s)

- (1) 181IT230 - Nirmal Khedkar 
- (2) 181IT220 - Jeeukrishnan Kayshya 

as the record of the work carried out by them, is *accepted as the B.Tech. Project Work report submission* in partial fulfillment of the requirement for the award of degree of **Bachelor of Technology** in Information Technology in the Department of Information Technology.

Guide

Dr. Ananthanarayana V. S.

Dr. Jaidhar C. D

Chairman - DUGC

ABSTRACT

Today there is still an unbelievable amount of enthusiasm for blockchain technology and cryptocurrencies. Decentralized networks like these are being hailed as solutions to countless challenges of trust and control.

While chain decentrality and TPS are a very hot topics in this field, we think that not enough research is being done in making blockchains more practical from a legal viewpoint. Blockchains and even crypto-currency blockchains like Bitcoin hold arbitrary information in addition to financial transactions. With the rise in the use of blockchains, we argue that this may create a problem: node operators (or aspirants) may want to be a partial node only (and not a full node), to not suffer legal ramifications if a blockchain transaction ever had inappropriate/ illegal data within it.

If a blockchain falls into legal trouble for its data, we find that there is not much it can do other than take drastic actions which would require a consensus of all full node operators, which in itself requires a lot of hard work and off-chain coordination. But because cryptocurrency universe is such huge in market cap (USD 2 trillion), not working on such legal implications of blockchain will only discourage full node operators, which is not very healthy for distributed systems like blockchains.

To combat this, we describe a general local erasure approach for UTXO-based (i. e., Bitcoin-like) cryptocurrencies for safely erasing transaction data from the local storage of these nodes. This way we will be able to locally erase data while keeping the remaining full node functions and benefits intact. We then extend the same to other popular blockchains like Litecoin or Bitcoin Cash. We would then also understand how local erasure can impact other chain operations like distributed mining, chain integrity and block integrity.

Keywords— blockchain, cryptocurrency, erasure, legality, node

CONTENTS

LIST OF FIGURES	i
LIST OF TABLES	ii
1 INTRODUCTION	1
1.1 Overview	1
1.2 Non-Financial Data on Blockchain	1
1.3 Motivation	2
2 LITERATURE REVIEW	4
2.1 Background and Related Works	4
2.2 Outcome of Literature Review	5
2.3 Problem Statement	5
2.4 Project Objectives	5
3 PROPOSED METHODOLOGY	7
3.1 Our Approach	7
3.2 UTXO-based Blockchain Protocols	7
3.3 Making deletion requests decentralized	8
3.4 Erasing Transaction Parts	8
3.5 Validating with Erased Data	9
3.6 Receiving Data and Bootstrapping New Nodes	9
3.7 Implementational Details	10
3.7.1 SegWit Support	10
3.8 Extendable to UTXO based blockchains	11
3.9 OP_RETURN Outputs	11
3.10 Constraints	12
4 RESULTS AND ANALYSIS	13
4.1 Proof of Concept in Action	13
5 CONCLUSION AND FUTURE WORK	16
5.1 Conclusion	16

5.2 Future Work	16
REFERENCES	17
A BIO-DATA	19

LIST OF FIGURES

3.5.1 Validations of incoming transactions [11]	9
4.1.1 Erasure-worthy blocks	13
4.1.2 Program logic flow	14
4.1.3 Sample Result of Proof of Concept (Bitcoin)	15

LIST OF TABLES

2.1.1 A review of existing literature in similar fields	4
---	---

Chapter 1

INTRODUCTION

1.1 Overview

Blockchain is a new technology that can allow users or groups to pay accounts between them without having to rely on affiliated organizations [7]. To maintain this distributed ledger containing transactions, it relies on cryptographic proof of the work, digital signatures, and contacts with the consulting providers like node and wallet operators.

Each block in the blockchain contains data such as facts or transaction details that are recorded with a cryptographic principle or a hash value. That hash value is composed of an alphanumeric string generated by each block separately. Every block contains a hash or digital signature for both itself and the preceding block. This ensures that the blocks are retroactively coupled together and unyielding. We can also ensure that no one can hack into the system or change the information stored in the block in this manner.

Each data block, such as facts or transaction details, continue to use this cryptographic policy or hash value. This hash value is made up of a string of numbers created independently by each block separately. All blocks contain not only the hash or digital signature itself but also the block preceding it. This ensures that the blocks are connected. This blockchain technology application ensures that no one can access the system or modify the data stored in the blockchain.

1.2 Non-Financial Data on Blockchain

While blockchains like Bitcoin were primarily intended for financial data, news and articles have documented that the public very quickly started using blockchain for digital arts, to verify ownership, timestamp record, etc.

Data on blockchains like Bitcoin can be stored as:

- Fake addresses: The text data is encoded and converted into a standard Bitcoin address. One then sends some Bitcoins to this address to have it recorded on

the chain. The bitcoins are lost (as the address is fake), but the data is recorded on the chain.

- Custom ScriptPubKeys: The text data is encoded and inserted into a standard ScriptPubKey. “ScriptPubKey” is a field in all Bitcoin transactions that allows users to run scripts on the node.

Blockchain developers (Bitcoin in particular) have agreed in principle that blockchain is a great way to record information that might be of significant importance or one which could otherwise be lost. But since the above methods add unspendable transactions onto the node memory, Bitcoin developers devised a new set of transactions called “OP_RETURN”, which are meant specifically for arbitrary data storage and are dealt with differently.

All these methods have been used extensively for harmless purposes over the years: transactions like 8881...2dba have encoded an image of Nelson Mandela and a quote from him. The first-ever transaction on Bitcoin has a reference to the global financial crisis in 2008.

However, this facility has been used for dodgy purposes too: to store things like firmware keys, illegal numbers, obscene information, trade secrets, etc. Holding such information would be illegal in many parts of the world. Segura and Sola’s research [9] into non-financial data on blockchain corroborates the same fact too.

As a result, an existing or potential operator nodes today faces the following options: deploy a complete node, contribute to the entire network’s stability, and store any illegal information that comes with it — or participate as a pruned node with reduced security and privacy.

1.3 Motivation

Because of the confusion and the mania around blockchain and its applications like cryptocurrencies, smart contracts and NFTs, very little in blockchain blockchain has never been studied from the lens of legality. While existing approaches have been used to address this issue, they are either not practical enough to be deployed, too slow or put too much trust into any one party (which defeats the point of Bitcoin’s decentrality). We believe that once the legal systems understand and extend today’s

laws to blockchains, blockchain community and its node operators would immediately be in trouble, because objectionable content has already been stored on the blockchain [2], and it is being distributed to different nodes as a part of typical blockchain operations.

This led us to framing our problem: we wanted to build a lightweight local framework around UTXO based blockchain software that is able to locally obscure (mask) arbitrary data without making significant changes to the chain's core code. This way our tool would be extendable, maintain chain integrity, maintain node synchronisation in this distributed environment and abide to laws.

Chapter 2

LITERATURE REVIEW

2.1 Background and Related Works

Table 2.1.1: A review of existing literature in similar fields

Paper Name	Brief Idea	Drawback
Matzut et al [1]	To detect the kind and content of data that goes on the blockchain and introduce protocol modifications to increase the cost of adding arbitrary data.	We believe that while content detection before insertion is a good idea, it does not really solve the problem of erasing the data that is already on the blockchain: if a bad actor circumvents this filter and adds objectionable data to the blockchain, we would again have the same problem. Also, increasing the cost to add arbitrary data may also reduce the huge scope of applications that blockchain systems have.
Ateniese et al. [4]	To replace the hashing used in blockchains with chameleon hash[5] functions, which would give trusted entities a trapdoor key to calculate hash collisions. This way they would be able to erase the data while maintaining the chain's integrity.	Implementing such solutions would result in a massive impact on the underlying trust model of blockchain systems.
Chepurnoy et al [10]	To store the global state of the chain separately so that the older blocks can be safely pruned.	While this solves the erasure problem for old blocks, it does not apply to new blocks: one would yet not be able to erase data from a (relatively) new block without impacting the chain's integrity.

2.2 Outcome of Literature Review

[1], [5] and [10] would require major changes to the core blockchain software and in some cases even a sense of trust between all nodes on the network.

We also review the research of Florian et al. [11], who propose a system wherein minor protocol changes can be made on the node software to erase the data as well as not impact the chain's integrity. One could do this by saving the erased data in a separate database, and not relaying confirmation for any transaction which depends on the said data. This way we can ensure that forks don't happen in the future. If transactions depending on erased data are included in the blockchain, the nodes can assume that the data has been erased successfully.

While such a system would work, we argue that such a system may not be compatible with blockchains other than Bitcoin (Bitcoin Cash, Litecoin), nor might it work to detect all kinds of transactions that may have objectionable data in it. [11] does not explain how OP_RETURN transactions are handled either.

2.3 Problem Statement

Build an efficient way to erase objectionable non financial data from blockchains which would make it easier for blockchain communities from a legal standpoint.

2.4 Project Objectives

- A practical and individually deploy-able approach to local erasure that addresses issues like the erasure of potentially consensus-critical data. [11]
- A proof-of-concept tool that shows how local erasure can be applied to existing Bitcoin nodes. [11]
- A detailed discussion of legal and non-legal reasons for local erasure, as well as the potential implications of local erasure when applied to existing networks [11]

- Validates the application of local erasure to other blockchain architectures, towards enabling local erasure in further popular networks such as Litecoin, Bitcoin Cash, etc.
- Understand and implement means to erase data in OP_RETURN or coinbase transactions, without harming chain integrity or node synchronisation.

Chapter 3

PROPOSED METHODOLOGY

3.1 Our Approach

In the approach, first a node with objectionable data make request to erase data from the blockchain. When the node raise a request all other trusted nodes go through voting process to decide whether or not to delete the requested data. Local erasure is the process of indentifying and erasing specific groups of information from the storage. They are stored in a format that cannot ever be reconstructed again (garbled/obscured). In essence, the data is deleted and saved in an erasure database, so that nodes would automatically avoid seeking the same in the future.

The following principles are used whenever validation steps depend on portions of erased data:

1. Unconfirmed transactions whose validity is based on erased data are ignored and never relayed by nodes.
2. If transactions based on deleted information are embedded in the blockchain, deleting nodes assumes that verification tasks related to deleted data are completed correctly.

3.2 UTXO-based Blockchain Protocols

We intend to investigate aspects of UTXO-based protocols relevant for the storage and erasure of objectionable data, specifically the definition of data structures and fields.

The main part of transactions is their inputs and outputs. Outputs encode cryptocurrency value units that can be spent, with a scriptPubKey encoding the necessary prerequisites. Inputs include a reference to previous transaction outputs as well as a solution to output scripts. Transaction outputs can only be used once in this manner, allowing them to be distinguished as spent and unspent transaction outputs, STXOs and UTXOs.

3.3 Making deletion requests decentralized

As we know, the blockchain actually contains two types of transaction one is confirmed and other is unconfirmed . When a transaction is created initially it is in unconfirmed state and a miner comes and confirmed it and put it in the blockchain. Now in this way the blockchain contains only confirmed transactions and nodes are also confirmed.

So now, when a node identifies some objectionable non financial data it will actually raise a request to delete the data. Now all other nodes in the chain which are also confirmed by the miner, will take part to decide whether we need to delete it or not . After going through a voting, if all nodes which are confirmed by miner thus can be trusted comes up with mutual agreement to delete the objectionable data, blockchain node operators can apply the algorithm to replace the objectionable data with garbage data.

3.4 Erasing Transaction Parts

As also explained in [11], we believe that a node operator can label transaction sections as deleted or obscured whenever considered necessary by the network. As a result of marking a part X of a locally stored transaction T as erased:

1. T can be stored in a separate database (which we can call “erasure database”), with X overwritten by substitute values, yielding T.
2. T can be physically erased or overwritten in such a way that it can’t be recovered. T is physically deleted or overwritten with T from its original storage locations, rendering it unrecoverable.
3. Any operation that relies on stored transactions should check the erasure database to make sure no relevant input data has been lost. The saved redacted transaction (such as T) is used for further operations if it has been wiped.

3.5 Validating with Erased Data

As described in [11], we cannot simply erase a UTXO because it opens up the danger that future blocks and transactions will not be deemed correct, leading to a fork. To avoid a fork, we propose to implement two simple rules:

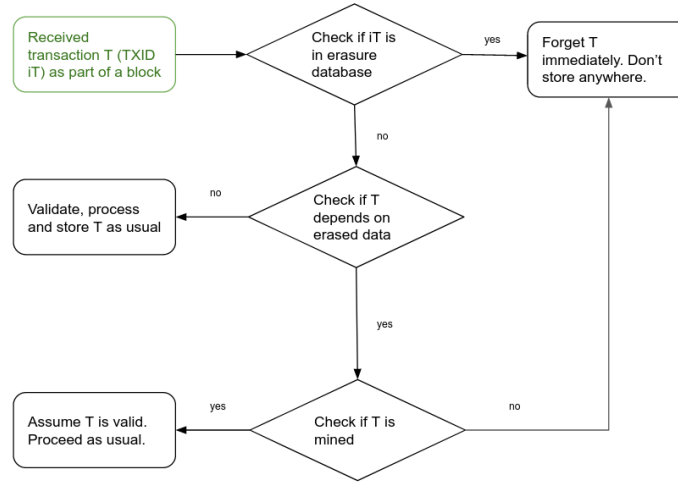


Figure 3.5.1: Validations of incoming transactions [11]

1. Any unconfirmed transaction referring erased data is always considered invalid and never relayed to other peers.
2. It is presumed that if a transaction was included in a block, it was deemed correct by at least one node operator, and that the transaction will therefore likely remain part of the chain.

3.6 Receiving Data and Bootstrapping New Nodes

While the above-lined approach works for old nodes, a new node operator may not be aware of the objectionable data they might store in their databases. We think this problem can be solved by two means:

1. Ask for the erasure database from one of the existing nodes.
2. Ask for the erasure database from a trusted third party.

As also explained in [11], We consider the second alternative to be slightly better because that way we are sure that the source of the database would leave the non-problematic outputs of T in their original form. Regardless, as future scope of this project, we would look into how such heavy dependence on an external trusted source also is minimised.

3.7 Implementational Details

On exploring Bitcoin's codebase and data folder structure, we can tell that Bitcoin stores blockchain data in the following folders:

1. blocks/: Raw block data in a custom format. Our tool should erase data from this location.
2. chainstate/: From v0.8, Bitcoin uses a LevelDB key-value database to store different aspects of blockchain like a copy of all current UTXOs. Our tool should erase the UTXO payload marked for erasure from here.
3. mempool.dat: Includes valid transactions not yet included on the active chain. These will be included in our chain soon.
4. General blockchain logs.
5. indexes/txindex/: This is an optional transaction index database.

We can ignore the final three remaining Bitcoin configuration files because they're not relevant for our problem statement. Because Bitcoin uses its own algorithms to build raw block data, we will use open source libraries to make RPC calls to Bitcoin software, to interact with the Bitcoin node software.

3.7.1 SegWit Support

One of the problems Bitcoins faced (and faces to this day) is the size crunch: Bitcoin blocks can have a maximum size of 1MB. Because of this, only a limited number of transactions can be added to a block.

Digital signatures take up 65 percent of a transaction's space. By extracting the signature from the input and relocating it to a structure near the end of the transaction, SegWit tries to ignore the data tied to a signature. This would increase the one MB limit for block sizes to a little under four MB. To do so, it adds a new witness structure to the mix, which must be provided in addition to scriptSig.

However, this will also be a hindrance to our erasure: when spending from SegWit UTXOs, witness data is bundled together with scriptSig. It cannot be influenced at the time of erasing.

We have concluded from our analysis that the only way we can overcome SegWit related challenges is by modifying validation rules in the chain core software while implementing a local erasure system. However, as one of our project objectives is to build this tool as an extension to popular blockchains and not make it specifically for any single chain, we would leave this under future improvements to this problem.

3.8 Extendable to UTXO based blockchains

While Bitcoin is a highly valued digital asset in itself, it is one of the blockchains with the least size on output scripts, which in turn means that any bad actor has much little flexibility writing arbitrary and illegal data on Bitcoin than what he/she has when writing it on other UTXO based assets. This is why we also wanted to build this tool in a way that it can obscure data in non-Bitcoin UTXO based chains too. While these blockchains are considerably different from Bitcoin, we find that there are some similarities in the way these are developed, which we can use for the benefit of our tool.

3.9 OP_RETURN Outputs

One cannot simply remove outputs from chainstate database, as it can have side effects like: coins can be burned or some of the outputs might be spendable or already spent. We prevent breaking the transaction integrity by only obfuscating or masking content from manipulated transactions, i.e., the content cannot be recovered after the redaction but any spendable output remains spendable. However, data inserted via

intended methods, i.e., `OP_RETURN` and `coinbase`, is not tied to other transactions. Such content can be removed without affecting the validation of other blocks.

3.10 Constraints

1. Because Bitcoin or Litecoin’s pruning capability erases all blocks up to a preset height, more data is removed than is required. As a result, our tool is currently only useful when full blockchain archival and indexing are not required. Notably, nodes continue to verify the entire blockchain.
2. Because we rely on pruning the nodes, full erasure of output can only take place when the node has been properly synced up to the point where the output is “buried” below 300 blocks, or about 50 hours after content has been added to the active chain.
3. SegWit UTXOs cannot be erased without making changes to core Bitcoin code. Thus, as mentioned above, SegWit based UTXOs will not be erased by our tool.

Chapter 4

RESULTS AND ANALYSIS

After understanding cryptocurrencies and the basics of UTXO blockchains like Bitcoin and Litecoin, we have worked on the removal of non-financial data from nodes and successfully we erased those non-financial data which is creating issues for privacy. Please find a detailed description of our proof of concept and our remaining work until the next evaluation below.

4.1 Proof of Concept in Action

We will use this tool to create a custom transaction onto the blockchain, and then obscure and prune the same transaction.

We record outputs to be erased as a JSON object. In a production environment, we expect this data to be stored or maintained by a trusted source.

```
{
  "info": {
    "00ba661f4592787128057ed7db5a3149b62214ba796bb77108b2dcf99ce033d2": {
      "0ec36fc785afccd5d9421fad5350c35c5edefe9b73cb1de6388fb7913d86233c": [
        0
      ]
    },
    "4b47cac91779a3fd8747f675dc53b63f4d9be00a99f8fceb4e3aff7f4be2f191": {
      "0fc2e717555f925e703b17a28ea92c31b078b125da61ac6fe5dd4924ac7adf1d": [
        0
      ]
    },
    "6a208774daae1d8aad85ffe2a64171c6ff867a9f0c60607154242d7a1057160c": {
      "bf3482b694a13b161a72e7f6537df0777da98e82124707c1cdba777b5d71905b": [
        0
      ]
    }
  }
}
```

Figure 4.1.1: Erasure-worthy blocks

Such object includes block ID, transaction ID and the index of the objectionable output.

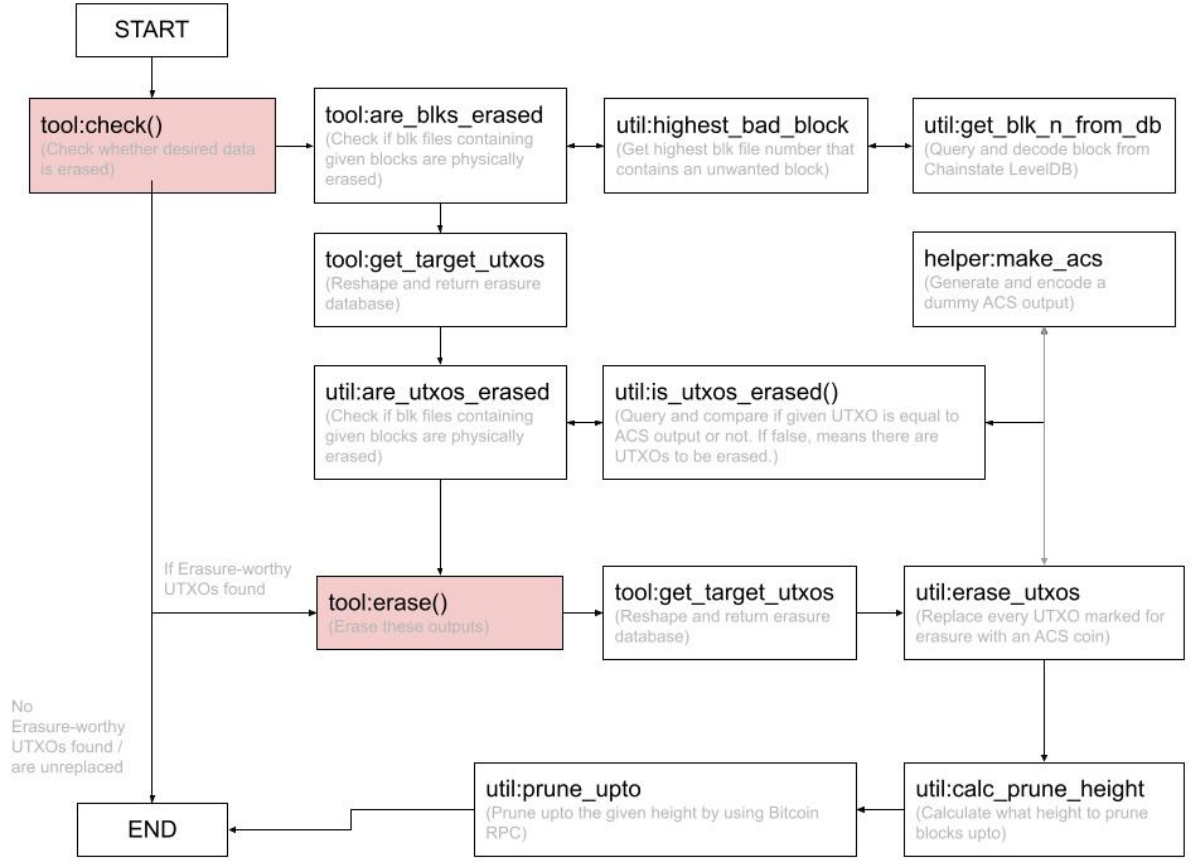


Figure 4.1.2: Program logic flow

Before the script begins, we must close any existing Bitcoin processes. This is because Bitcoin uses a LevelDB database which creates a lockfile every time data is being read, because of which two programs cannot read the same data at the same time.

The crucial steps in the program are highlighted in red. We first scan the existing files in node directory to check if there are any unwanted transactions (*tools : are_blks_erased*). We do this by first checking the files that contain those blocks physically (*util : get_blks_n_from_db*). If no such files are found, we look into the chainstate database (which holds UTXO information) for the listed transactions and their outputs.

If any blocks or transactions are found, we first obscure them from the chainstate database by replacing them with dummy P2TR outputs (*util : erase_utxos*). After

this, we restart the Bitcoin node and prune the blocks database up to the given height (*util : prune upto*). The pruning for the same is performed by making an RPC call to the node process. This way, a bitcoin UTXO output is permanently obscured.

```

nimal@legend:~/Documents/DevWorld/IT/Projects/HP/projectrepo
$ ./src/tool.py
('00ba061f4592787128057ed7d85a3149b62214ba796bb77108b2dcf99ce033d2': {'0ec36fc785afcc5d9421fad5350c35c5edfe9b73cb1de6388fb7913d86233c': 0}), {'4b47cac91779a3f68747f675dc53b63f4d9be0e099f8fceb43aff774be2f191': 0}), {'0fc2e717555f925e78b317a28ea92c31b078b125da61acfe5d64924ac7adfd1d': 0}), {'6a208774dca01d8a085ff62a64171cdf867a9f6c0e06715424267a1857160c': 0}), {'bf3482b694a13b161a72e7f6537df677da98e82124707c1cda777b5d719b5b': 0})])
Please make sure the corresponding bitcoin instance is stopped. (Press Enter.)
Processing blk 00ba061f4592787128057ed7d85a3149b62214ba796bb77108b2dcf99ce033d2
Processing blk 4b47cac91779a3f68747f675dc53b63f4d9be0e099f8fceb43aff774be2f191
Processing blk 6a208774dca01d8a085ff62a64171cdf867a9f6c0e06715424267a1857160c
Processing blk 00ba061f4592787128057ed7d85a3149b62214ba796bb77108b2dcf99ce033d2
Processing blk 4b47cac91779a3f68747f675dc53b63f4d9be0e099f8fceb43aff774be2f191
Processing blk 6a208774dca01d8a085ff62a64171cdf867a9f6c0e06715424267a1857160c
Lowest stored block number: 2
Highest 'bad' block number: 0
Checking UTXO ('0ec36fc785afcc5d9421fad5350c35c5edfe9b73cb1de6388fb7913d86233c', 0)
UTXO ('0ec36fc785afcc5d9421fad5350c35c5edfe9b73cb1de6388fb7913d86233c', 0) is not erased
Checking UTXO ('0fc2e717555f925e78b317a28ea92c31b078b125da61acfe5d64924ac7adfd1d', 0)
UTXO ('0fc2e717555f925e78b317a28ea92c31b078b125da61acfe5d64924ac7adfd1d', 0) is not erased
Checking UTXO ('bf3482b694a13b161a72e7f6537df677da98e82124707c1cda777b5d719b5b', 0)
UTXO ('bf3482b694a13b161a72e7f6537df677da98e82124707c1cda777b5d719b5b', 0) is not erased
Some unwanted transaction outputs are stored locally.
Will erase locally, editing bitcoin data files. (Press Enter.)
Replacing target UTXOs with 'anyone-can-spend' outputs.
Erasing 0ec36fc785afcc5d9421fad5350c35c5edfe9b73cb1de6388fb7913d86233c:0
Erasing 0fc2e717555f925e78b317a28ea92c31b078b125da61acfe5d64924ac7adfd1d:0
Erasing bf3482b694a13b161a72e7f6537df677da98e82124707c1cda777b5d719b5b:0
Getting height to prune to.
Processing blk 00ba061f4592787128057ed7d85a3149b62214ba796bb77108b2dcf99ce033d2
Processing blk 4b47cac91779a3f68747f675dc53b63f4d9be0e099f8fceb43aff774be2f191
Processing blk 6a208774dca01d8a085ff62a64171cdf867a9f6c0e06715424267a1857160c
Please edit your node's configuration to enable pruning ("prune=1") and start your node. (Press Enter when ready.)
Pruning blk files upto height 4
Erase complete! (Your node is still running.)
nimal@legend:~/Documents/DevWorld/IT/Projects/HP/projectrepo

```

Figure 4.1.3: Sample Result of Proof of Concept (Bitcoin)

Above is the sample output of our proof of concept. Notice that for our demonstration purpose, we query the nodes to be deleted from a dummy server (which in the production environment we expect the government or the Bitcoin community to handle).

This way, we are able to locally erase problematic data in blockchain nodes without using content detectors, redactable blockchains or chameleon hash functions, all of which are either very impractical or jeopardise the trustless nature of blockchain. When in future, the world's court extend the laws of data storage and distribution to blockchain nodes, our solution of local erasure would help full node operators comply with the rules quickly, giving them no legal hassles and very little management overhead.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this paper, we come up with a local erasure solution from the blockchain. We argue the common idea that erasure at any level is not possible at all in blockchain and what are the implications of not erasing some transactions in popular blockchains like Bitcoin.

We can conclude that by doing a local erasure we enable blockchain networks to embrace a larger range of legal norms and individual values, clear any chance of any legal hurdles for node operators and therefore become truly global networks.

5.2 Future Work

Currently, we completed our work on bitcoin blockchain and extended it to another blockchain called litecoin. There are many more blockchains available which can be tested with our implementation. But as of now our approach is suitable for UTXO based blockchain only. So for further step ,this approach can be extended to wallet based blockchain and try our implementation part on such blockchains.

REFERENCES

- [1] R. Matzutt, M. Henze, J. H. Ziegeldorf, J. Hiller, and K. Wehrle, “Thwarting unwanted blockchain content insertion,” in 2018 IEEE International Conference on Cloud Engineering (IC2E)
- [2] R. Matzutt, J. Hiller, M. Henze, J. H. Ziegeldorf, D. Müllmann, O. Hohlfeld, and K. Wehrle, “A quantitative analysis of the impact of arbitrary blockchain content on Bitcoin,”
- [3] E. Lombrozo, J. Lau, and P. Wuille, “BIP141: Segregated witness (consensus layer),” 2015. [Online]
- [4] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, “Redactable blockchain—or–Rewriting history in Bitcoin and friends,”
- [5] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, “Chameleon-hashes with ephemeral trapdoors,” in IACR International Workshop on Public Key Cryptography
- [6] M. Berberich and M. Steiner, “Blockchain technology and the GDPR– how to reconcile privacy and distributed ledger”
- [7] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Oct. 2008.
- [8] European Union, “Directive 2000/31/EC of the European Parliament and of the Council of 8 June 2000 on certain legal aspects of information society services, in particular electronic commerce, in the Internal Market (‘Directive on electronic commerce’).” [Online]. Available: <https://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX3A32000L0031>
- [9] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera- Joancó-martí, “Analysis of the Bitcoin UTXO set,” in Financial Cryptog- raphy and Data Security
- [10] A. Chepurinov, M. Larangeira, and A. Ojiganov, “Rollerchain, a blockchain with safely pruneable full blocks,”

- [11] Martin Florian, Sophie Beaucamp, Sebastian Henningsen and Bjorn Scheuermann, “Erasing Data from Blockchain Nodes”.

Appendix A

BIO-DATA

Name: Nirmal Khedkar
Address: National Institute of Technology Karnataka, Surathkal
Email: nirmal.181it230@nitk.edu.in
Contact No.: +91-9356649190

Name: Jeeukrishnan Kayshyap
Address: National Institute of Technology Karnataka, Surathkal
Email: jeeukrishnan.181it220@nitk.edu.in
Contact No.: +91-9101265983