Decentralised Redactable Blockchain: A Privacy-Preserving Approach to Addressing Identity Tracing Challenges

Anonymous Submission

***Abstract*—Blockchain is an immutable and distributed ledger managed by all participants, enhancing data transparency and safety. Immutability is a crucial factor in ensuring data transparency and safety. However, there is a significant demand for redaction of the ledger due to security and privacy concerns. For instance, service providers have to comply with strict requirements for handling personal data under the European General Data Protection Regulation (GDPR). Bitcoin and Ethereum suffer from the presence of harmful, illegal or privacy-sensitive data within the ledger. In this paper, we propose a retractable blockchain solution based on meta-transaction using zk-SNARK to improve anonymity in a decentralised manner. A one-time cryptographic key generation scheme for a signature generation scheme, generates different keys for each transaction to enhance security and privacy by preventing identity tracing. We also employ zk-SNARK to hide the information of cryptographic keys and signatures. This approach enforces anonymity, ensuring that transaction owners, who are exclusively authorised to modify their transactions, remain undisclosed. The records of modification history through meta-transactions are preserved in the blockchain ledger. The modification history for each transaction is linked together, and the verification time is significantly short, around 10 milliseconds, even when transactions have multiple modifications. Furthermore, we introduce a redaction fee scheme for transaction owners to maintain concise modification histories encouraging removal instead of modification. This modification fee scheme can minimise the performance overhead associated with this redaction approach.**

***Index Terms*—redactable blockchain, blockchain redaction, zk-snarks**

I. Introduction

As blockchain is a distributed ledger shared by all participants, its unique and outstanding features are immutability and decentralisation. The chained architecture, which links blocks with cryptographic hashes, ensures the immutability of the blockchain. Additionally, storing a copy of the ledger in participants' local storage makes blockchain decentralised. Blockchain's transparency and auditability are a result of these features, but they also lead to scalability, privacy and security issues.

There is a huge demand for redactable blockchain against immutability. The European General Data Protection Regulation (GDPR) imposes strict requirements on service providers for handling personal data, including the "right of rectification" and "right to be forgotten" [1]–[4]. Another issue arising from the immutability of blockchain is the unintentional or intentional storage of privacy-sensitive or harmful data [5]–[10]. For example, according to [11], Bitcoin contains more than eight files with illegal and condemned content, and seven files violate copyrights. Users may want to delete their personal data and the system may also need to remove harmful data from the blockchain not only to save the storage space but also to improve security and privacy.

Moreover, the rapid growth of the ledger and the tremendous demand for data purging against blockchain immutability have led to active research on redactable blockchain. As of 2023, the ledger sizes of Bitcoin and Ethereum have reached 500 and 700 GB, respectively, and they continue to grow over time [12].

Several researchers have proposed redactable blockchain solutions not only to enhance security and privacy but also to reduce blockchain size. The work in [13] proposes a redactable blockchain solution based on chameleon hash, which allows for finding arbitrary collisions with a trapdoor key. This chameleon hash-based solution provides flexibility for modifying, deleting or inserting data without breaking the hash consistency. However, it demands tremendous computational power. Additionally, the solution can compromise decentralisation and security since the trapdoor key must be kept secret. [14] introduces a novel redactable blockchain solution based on a polynomial calculation using Lagrange interpolation. In this solution, a block includes coordinates of transactions and a unique polynomial computed based on these coordinates, but it does not include a hash of its previous block in the block header. This design ensures that blockchain consistency can be maintained even when the block is modified. [15] proposes a redactable blockchain solution based on the Rivest–Shamir–Adleman (RSA) algorithm. In this approach, instead of using cryptographic hashes to connect blocks, each block is linked to its previous block using a prefix that is computed based on a secret key and public key, similar to how RSA operates. By employing this technique, a central authority possessing the secret key can modify blocks without breaking the consistency of the blockchain. MOF-BC is a redactable blockchain solution based on meta-transaction [16]. With this approach, by issuing modification transactions known as meta-transactions, transaction owners can remove, summarise and age their transactions that have been committed to the ledger. To preserve hash consistency, the calculation of block hash in MOF-BC includes only transaction hashes.

As redaction can compromise the integrity of blockchain, redactable blockchain solutions must address tampering attacks. These attacks involve tampering with data through

TABLE I
SUMMARY OF REDACTION SOLUTIONS

Works	Redaction	Security assumption	Anti-Tampering attack	Anonymity solution	Authentication risk	Central Authority
Chameleon hash-based [13], [17]	Block	Chameleon hash function	Key-based	No	Key exposure [13]	Key management [13]
Polynomial-based [14]	Both	Polynomial	Polynomial-based	No	-	Modification permission
RSA-based [15]	Block	RSA	Key-based	No	Key exposure	Key management
MOF-BC [16]	Transaction	Certificate Authority	Key-based	One-Time Key	Key exposure	Certificate Authority
μchain [18]	Transaction	AES	Mutability policy	No	Key exposure	-
Proposed solution	Transaction	EdDSA zk-SNARK	zk-SNARK-based	One-Time Key	-	-

redaction without permission or using unpermitted methods [5], [19], [20]. As shown in Table I, each solution prevents tampering attacks with its own method to preserve blockchain consistency. However, many solutions require a central authority to manage cryptographic keys or modification permissions, which poses a potential risk of exposing these keys [5], [8]. This results in authentication risk concerning the management of the cryptographic key or redaction permission when accessed from unauthorised entities. Anonymity is also one of the most concerning issues regarding decentralised redaction solutions in blockchain based on cryptographic signatures or addresses.

We propose a novel solution for redacting blockchain using a one-time cryptographic key generation and zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARK) [21], [22]. Redactable transactions are signed with the Edwards-Curve Digital Signature Algorithm (EdDSA) [23] and then converted into proofs of zk-SNARK to validate permission for modification. To sign transactions, we apply a one-time key generation method, making it difficult to trace the transaction owners. Additionally, employing zk-SNARK can hide the public key and signature, enhancing anonymity, and the proofs can be verified by anyone in a decentralised manner. Each transaction can be modified by issuing redaction transactions, also known as meta-transactions, which indicates the transaction to be modified and includes the updated information. After the redaction transactions are mined and broadcasted by miners, the storage nodes update their ledger in the local storage when they receive a block, including the redaction transactions.

zk-SNARKs are known to require a trusted setup by central authorities [24]–[26]. However, there are many emerging solutions that do not require a trusted setup and are conducted in a decentralised way. For example, a Multi-Party Computation (MPC) protocol [27], [28] performs the setup process by multiple participants in a decentralised way. Spartan [29] introduces the first zk-SNARK solution without trusted setup. In this paper, we assume that the trusted setup process of zk-SNARK has been conducted in a decentralised way. The contributions of this paper are the following:

- A fully decentralised redactable blockchain solution based on meta-transactions. Since we utilise zk-SNARK for redaction of the blockchain ledger, there is no need of a central authority for cryptographic key management or modification permission. This solution can be applicable to any public blockchain.
- Enhanced anonymity by employing a one-time cryptographic key scheme and utilising zk-SNARK for each redaction transaction. To generate a signature for each transaction, the one-time cryptographic key is derived based on the concatenation of a transaction and a master key through a deterministic key generation method. Note that the master key is unique for each node. Additionally, we hide the exposure of the one-time key and signature in a transaction by using zk-SNARK, which has fast verification time and a small size of proof.
- Fraud resistance is achieved through a chained structure of transaction modification history. The transaction modification history is recorded in the ledger, linking each modified transaction to its corresponding modification transaction. This ensures that fraudulent activities, such as double-spending, can be easily detected by examining the modification history.
- Performance evaluation of the verification process in relation to the transaction modification history. Our solution shows remarkable verification performance even in the presence of a long modification history. Additionally, we introduce the concept of imposing redaction fees. This strategic measure aims to incentivise users to maintain their modification history effectively, minimising the performance degradation.

II. RELATED WORK

A. Chameleon hash-based

Chameleon Hash is a special cryptographic hash function with a trapdoor key that can compute hash collisions [30]. This feature allows for modifications to blockchain data while maintaining hash consistency. [13], [17] propose redactable blockchain solutions based on the chameleon hash, which can rewrite or remove blocks. These solutions utilise a chameleon hash function as an inner hash function to link blocks in the blockchain and add a randomness field is introduced in the

block header. To redact a block, the randomness field is updated using a collision calculated with the trapdoor key, which preserves the blockchain's consistency. The trapdoor key can be managed by a centralised auditor or shared among several parties. Only central authorities possessing the trapdoor key can redact or remove blocks in the blockchain. A new redaction block, generated by the central authorities, is broadcasted to all participants in the network. When nodes receive the new modified block, they can verify it and update their local ledger. These solutions employing the chameleon hash provide great flexibility for modifying, deleting or inserting data while preserving the hash consistency. However, these approaches demand tremendous computational resources for calculating hash collisions. Furthermore, the centralised key management system for the trapdoor key in [13] can potentially compromise authentication and integrity concerns due to key exposure [5].

B. Polynomial-based

[14] introduces a polynomial-based redactable blockchain structure aimed at eliminating fraudulent transactions. The key idea relies on the utilisation of Lagrange interpolation, which is an algorithm to construct a unique polynomial of N-1 degree, given N coordinates. Specifically, a unique polynomial is generated for each block using label fields found in both the block header and block body, which serve as the coordinates. The block body contains multiple transactions and is organised using the Lagrange interpolation method. To achieve this, each transaction and block header are transformed into coordinates using a mapping algorithm called hash-cut mapping, which converts strings to integers. The polynomial function assigned to each block represents its block address, and each block establishes a connection to its previous block using the polynomial function stored in its block header. Furthermore, a mechanism called modification difficulty is employed to adjust the computational cost required to modify a block in order to encourage users to accurately record data and reduce the frequency of modification operations. When users intend to modify blocks, they must recalculate the coordinates that satisfy the unique polynomial along with the associated difficulty for each block.

The polynomial-based approach provides the advantage of enabling redaction and deletion of blocks, while the adjustable modification difficulty offers flexibility to diverse security requirements and scenarios. However, due to the removal of the modification history, this approach can compromise auditability. Moreover, setting a high modification difficulty can lead to substantial modification costs, while a low modification difficulty may compromise the security of the system. In the context of the Proof of Work (PoW) consensus mechanism, this approach is not well-suited as modification can lead to a high rate of forks, causing potential disruptions to the blockchain's stability and reliability [8]. Additionally, managing modification permissions in this approach necessitates the implementation of additional security features such as authentication or supervision. However, incorporating these security measures may introduce centralisation tendencies, undermining decentralisation, which is the core principle of the blockchain system.

C. RSA-based

[15] proposes an innovative redactable blockchain method designed specifically for private blockchains, relying on the computational hardness of the RSA problem. Each block in this approach is composed of three components: a prefix P_i, the actual content C_i and a suffix X_i. To create a block B_i, $i-th$ block, a miner randomly selects a value for X_i from the range 1 to n-1, ensuring that X_i^2 is not 1 (mod n). Subsequently, a hash value h_i is computed by applying a hash function to the concatenation of P_i and C_i, represented as $h_i = HASH(P_i, C_i)$. The resulting hash h_i is then converted into an integer d_i (mod n). The prefix P_{i+1} of the next block is derived using the equation $P_{i+1} = (X_i)^{d_i}$ (mod n). To modify a block B_i, the central authority, possessing the private key, computes a new hash value $h_i' = HASH(P_i, C_i')$, where C_i' is the updated content. The new hash h_i' is then converted into an integer d_i' (mod n), ensuring that d_i' is relatively prime to $\phi(n)$, where $\phi(n)$ represents the Euler function of n. Once d_i' is determined, the central authority can calculates the inverse e_i' of d_i' (mod $\phi(n)$) and derives $X_i' = P_{i+1}^{e_i'}$ (mod n). According to RSA scheme, the integrity of the updated block can be verified through the equation $(X_i')^{d_i'} = (P_{i+1}^{e_i'})^{d_i'} = P_{i+1}$ (mod n).

The RSA-based approach offers a redaction method with strong corruption resistance, utilising the RSA scheme with low computational overhead. However, as this approach is specifically designed for private blockchains, a central authority is required to execute the redaction process. Consequently, the involvement of a central authority compromises the principle of decentralisation. Furthermore, the removal of modification history can raise concerns about the auditability of the blockchain [8]. For example, malicious attackers can exploit this limitation by easily reverting a block to its previous state since all versions can successfully pass validation.

D. Meta-transaction

Memory Optimised and Flexible Blockchain (MOF-BC) [16] introduces a memory-optimised and flexible approach for IoT users and service providers to perform data removal, memorisation and ageing within the blockchain. Unlike conventional blockchains that maintain immutability, MOF-BC addresses the challenge of data redaction by constructing block hashes based on transaction hashes rather than transaction content. This allows transaction content to be redacted while preserving hash consistency. A transaction generator can remove its transactions by issuing a removal transaction, called meta-transaction, which indicates transactions to be removed. Miners verify the hashes and the key in the removal transaction through the Generator Verifier (GV) to ensure the validity of the modification permission. The summarise method is designed to reduce the transaction history of asset transfers, particularly for transactions similar to Bitcoin. This method

involves summarising multiple transactions into a single consolidated transaction, which is achieved by issuing a summary transaction. On the other hand, the ageing method provides a way to compress the stored data based on the content type, specifically for IoT blockchains. In this method, the user, who is the generator of the original transaction, can compress the data using various compression techniques, including lossy or lossless compression [31]. Additionally, MOF-BC incentivises users to manage their data by offering a reward mechanism for data removal, Summarising or ageing. To facilitate these functionalities, MOF-BC adopts a shared read-only central database known as the blackboard, which manages the removal history and reward information.

By issuing meta-transactions, MOF-BC introduces three methods to modify blockchain data. However, the verification process is vulnerable, creating a potential risk of double-spending that can compromise integrity [5]. Moreover, when the content is compressed using lossy compression methods, it becomes challenging to verify whether the aged transaction is generated from its original transaction. Additionally, MOF-BC utilises a blackboard approach and CA, which act as centralised components.

μchain [18] is a redaction solution designed to handle vulnerable smart contracts or remove abusive content from the blockchain. This approach maintains alternative versions of data records represented by transaction sets, containing possible transaction versions to update transactions. Within a transaction set, only one transaction is active, while the others remain inactive alternatives. To extend a set of transactions, extended transactions referencing their mutable counterparts can be issued. Legitimate users specified in the transaction policy can update an active transaction by issuing a mutant transaction that defines a new active transaction. Validators verify that the transaction policy has been met during the mining process. Since mutant transactions can be issued multiple times, μchain proposes a transaction tree structure to efficiently manage transaction history. To enhance confidentiality, μchain encrypts transactions in mutable transaction sets using transaction-specific keys. Encryption with Secret Sharing (ESS) is employed to manage these keys, which are divided into shares and distributed among validators. Reconstruction of the keys is possible only when enough shares have been collected.

However, μchain has a limitation in that it does not delete inactive transactions from the blockchain record; instead, it encrypts them to hide their visibility. Consequently, this approach does not improve storage efficiency. Additionally, the use of encryption requires extra computational resources and raises concerns about auditability [20]. Moreover, enhancing confidentiality through encryption introduces the risk of key exposure, potentially compromising authentication and integrity.

III. The modifiable blockchain

A. Redactable transaction

1) Structure: The structure of a redactable transaction is designed for removal or modification of transactions by the transaction owner. Transaction owners have permission to modify their own transactions and they also have the responsibility for managing their cryptographic keys. As shown in Figure 1, the redactable transaction consists of a message, a one-time public key for the message, the previous transaction ID, and a proof for the message. EdDSA is utilised to sign the message, but note that other cryptographic algorithms are also possible, such as Elliptic Curve Digital Signature Algorithm (ECDSA) [32]. In this paper, SHA-256 is utilised to calculate hash values for deterministic random Oracle [33].

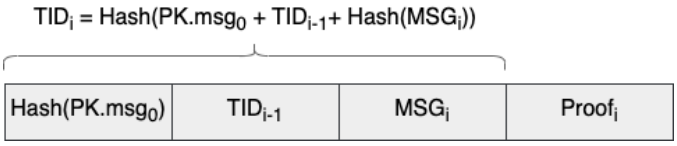

Fig. 1. Structure of redactable transaction

2) One-time key: Each node has their own master key which is a random seed to generate a one-time cryptographic key to sign each message in transactions. We utilise EdDSA for a signing algorithm, which is used in many security protocols such as OpenSSH or transport layer security (TLS) [34], [35]. The one-time cryptographic key is generated from a master key and a hash of a message in a transaction. First, a one-time secret key, $SK.msg$, is derived as:

$$SK.msg = Integer(Hash(MK + message)) \cdot B \quad (1)$$

Where MK is the master key and B is the base point of EdDSA. Since the hash value is calculated from the concatenation of the master key and the message, $SK.msg$ is deterministic random. The one-time public key, $PK.msg$, corresponding to $SK.msg$, is generated according to [23]. Transaction owners do not need to store the one-time keys as the keys are deterministic, which leads to a reduced risk in managing cryptographic keys and easily reconstructs them.

3) Hash of transaction: TID_i is the hash value of the $i-th$ transaction where i represents the modification history of the transactions. For example, $i = 0$ represents the original transaction. TID_i is calculated as:

$$TID_i = Hash(PK.msg_0 + TID_{i-1} + Hash(MSG_i)) \quad (2)$$

Where TID_{i-1} represents the previous transaction ID, which corresponds to the transaction modified by this current transaction. In the case of the first transaction,TID_0 has no previous transaction, so its previous transaction ID is 0. Transaction ID includes the one-time public key, its previous transaction ID and the hash value of the message, excluding the proof. The block header only includes this TID for each

transaction to preserve the hash consistency when a transaction is modified.

4) Proof: The proof of zk-SNARK for validation of modification permission is generated with a signature generated by EdDSA, the hash of $PK.msg_0$ and the hash of msg_i. zk-SNARK is known for its small size of proof and fast verification time without revealing any information [21], [36]. Since the signature is generated by EdDSA using $SK.msg_0$, the signature can be verified by corresponding $PK.msg_0$ and MSG_i by zk-SNARK in a decentralised manner without revealing $PK.msg$ and the corresponding signature. For the verification of modification permission, verifiers only need the hash of $PK.msg_0$, the hash of MSG_i and the proof of zk-SNARK. This can enhance anonymity for transaction owners and authentication for cryptographic keys. The process for zk-SNARKs is as follow:

- $Setup(C, \lambda) \rightarrow p_k, v_k$: The setup process is conducted just one time and p_k, v_k and C are opened to public. For example, the genesis block contains this information, where λ is a security parameter and p_k is a proving key, v_k is a verification key and C is the logical statement to be verified. In this paper, C verifies an EdDSA signature.
- $Proof(p_k, Hash(MSG_i), Hash(PK.msg_0), Sign_i) \rightarrow \pi$: $Sign_i$ represents EdDSA signature for the $i-th$ message as a secret input. The hash of a message and the hash of the public key for EdDSA are only opened to the public as public inputs. π is a proof of zk-SNARK.
- $Verify(v_k, Hash(MSG_i), Hash(PK.msg_0), \pi) \rightarrow 0/1$: As verfiers can verify proofs with public inputs, the hash of a message and the hash of public key for EdDSA, we can improve security and privacy.

B. Operation

1) Transaction creation: Figure 2 shows the processes to create a redactable transaction. Beside a user's role in creating a redactable transaction, the roles of miners and storage nodes rarely change. The steps involved in the creation and handling of redactable transactions are as follows:

- A user, the transaction owner, generates a one-time cryptographic key by randomising the seed through concatenation of their master key and the message (Step 1). The user generates a signature using the EdDSA algorithm with the one-time key (Step 2) and creates a proof of zk-SNARK to construct a redactable transaction (Step 3, 4). Subsequently, the user broadcasts the transaction to the blockchain network (Step 5).
- When miners receive transactions (Step 6), they verify the transactions (Step 7)and create a new block (Step 8), which includes TID for redactable transactions. The miners then broadcast the new block to the network (Step 9).
- Storage nodes store the new block in their local storage when they receive it (Step 10).

2) Transaction redaction: Figure 3 shows the processes to modify a redactable transaction. Note that the modification features include removing or updating a message in a

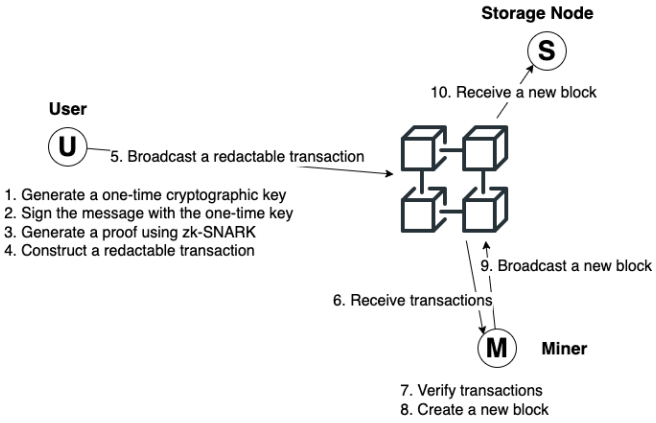

Fig. 2. Processes of creating a redactable transaction

redactable transaction. The steps involved in the modification and handling of redactable transactions are as follows:

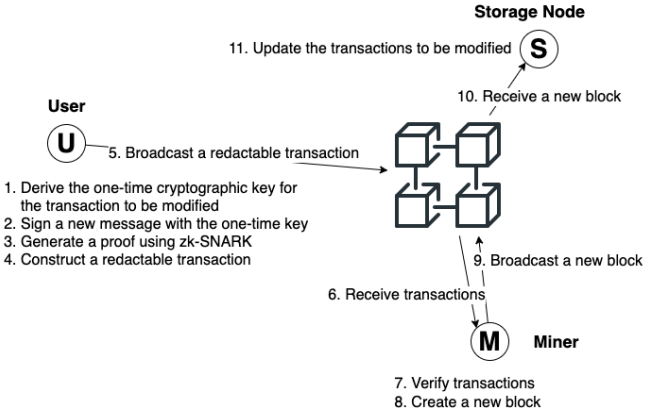

Fig. 3. Processes of modifying a redactable transaction

- The transaction owner easily derives the one-time cryptographic key, which was used when they signed the original transaction to be modified, by using their master key and the hash of the original message (Step 1). Only the transaction owner knows $SK.msg_0$ and can correctly sign the updated message with the secret key. Therefore, the modification permission can be validated by the same $PK.msg_0$ and the signature. The owner generates a signature using the EdDSA algorithm with the one-time key (Step 2) and creates a proof of zk-SNARK to construct a redactable transaction (Step 3). In this case, the previous transaction ID is the transaction ID to be modified (Step 4). Subsequently, the owner broadcasts the transaction to the blockchain network (Step 5).
- When miners receive transactions (Step 6), they verify the transactions (Step 7) and create a new block (Step 8), which includes TID for redactable transactions. For verifying the redactable transactions, miners may need to check additional conditions as well as the modification

permission. For example, in the case of cryptocurrency, miners must verify whether the transaction to be modified has been spent or not. The miners then broadcast the new block to the network (Step 9).

- Storage nodes update the transactions to be modified and store the new block in their local storage when they receive it (Step 10, ,11). The transaction to be modified indicates the transaction to modify, and vice versa as shown in Figure 4. $Transaction_i$ is a transaction to be modified and $Transaction_{i+1}$ is a new transaction to modify $Transaction_i$. To update $Transaction_i$ to be modified, storage nodes replace the $Proof_i$ field with the transaction ID to modify and replace the MSG_i field with $Hash(MSG_i)$. Since the Merkle root only includes the hash of $PK.msg_0$ and TID_{i-1} and the hash of MSG_i, the block consistency is not broken after modification. For ensuring finality, we recommend performing the update after the period of finality, such as 6 blocks for Bitcoin following the longest chain rule [37]–[39].

3) Transaction verification: As shown in Figure 4, we designed a chained architecture to manage the history of transaction redaction, where a transaction to be modified and a transaction to modify refer to each other. To verify the latest transaction in a chain, verifiers are required to retrieve the transaction history from the last to the first one, similar to the verification of Unspent Transaction Outputs (UTXO) in Bitcoin. In conventional blockchain systems, transaction verification is normally performed by full nodes who have an entire ledger of blockchain, so the process of retrieving transaction history only involves accessing the local storage. Figure 5 demonstrates the flow of verification of modification history of transactions, with the verification starting from the last transaction, which only has a proof within its modification history chain. The hash of $PK.msg_0$, TID_{i_1} and the hash of MSG_i can be easily verified by verifying the transaction ID in the Merkle root. On the other hand, it is necessary to verify TID_{i+1}, indicating the modifying transaction, by validating the modifying transaction. The modifying transaction must refer to the previous transaction to be modified and have a correct proof of zk-SNARK if it is the last transaction. Otherwise, it must refer to its modifying transaction.

4) Redaction fees: This redaction protocol leaves the modification history of transactions in the form of a chain linked to each other, allowing verifiers to easily retrieve the modification history. However, the long chain of history can compromise the performance of transaction verification. To maintain the chain short, we utilise transaction fees where the modification is significantly more expensive than removal. Note that redaction transactions are generally more expensive than creating new transactions. This differential rate of transaction fee scheme will encourage users to remove their transactions and issue new transactions instead of updating their existing transactions. Additionally, once transactions are removed, they cannot be updated anymore since users can easily issue new transactions.

The redaction solutions for blockchain have the potential to enhance privacy by removing or updating privacy-sensitive data from the blockchain ledger. However, it is important to note that the possibility of tracing the identification of modifiers during redaction can potentially undermine privacy. Similarly, these solutions can improve security by removing or updating vulnerable content within the blockchain ledger but malicious attackers can exploit the redaction process, potentially compromising security. Therefore, in the subsequent section, we will present a qualitative analysis of the privacy and security aspects of the proposed solution.

IV. Privacy and security

A. Anonymity

Using the same public key for verifying modification permission can compromise privacy, as the identity can be traceable [40]–[42]. To address this issue, we employ a one-time cryptographic key derived from the hash of the concatenation of the master key and a message. As a result, each message is signed with a unique cryptographic key, making it difficult to track user identity. Furthermore, to enhance privacy, the information of the key and signature are converted into a proof of zk-SNARK. This process ensures that transactions only include the hash of a public key and a proof of zk-SNARK, effectively hiding the actual public key and its corresponding signature. This approach provides an additional layer of privacy protection for users.

B. Authentication threats

Attackers may attempt to remove or modify transactions generated by other nodes, especially by forging the identity credentials [43]. The modification permission is only granted to the transaction owner who originally generates the transaction. To prevent unauthorised users from deleting or modifying others' transactions, we employ zk-SNARK as an anonymous authentication mechanism. zk-SNARK allows miners to verify the modification permission without revealing any information about the transaction owners. They can simply check a proof of zk-SNARK in a modifying transaction to ensure its validity. Furthermore, we implement the one-time cryptographic key scheme to address the private key leakage problem [44], [45]. Also, by hiding the key and signature through zk-SNARK, we ensure that sensitive information remains secure and inaccessible to potential attackers.

C. Key management

Many redaction approaches based on cryptographic algorithms carry the risk of exposing keys and also pose difficulties in managing the keys securely, as shown in Table I. However, with the one-time cryptographic key scheme and zk-SNARK, each node only keeps their master key secret, without requiring any other authorities to manage their keys. Since the one-time keys are also deterministic, each node does not need to store the cryptographic keys that they have used to generate their transactions. As a result of these measures, we effectively eliminate the key management problem in a decentralised manner, providing a robust solution to redact blockchain ledger.

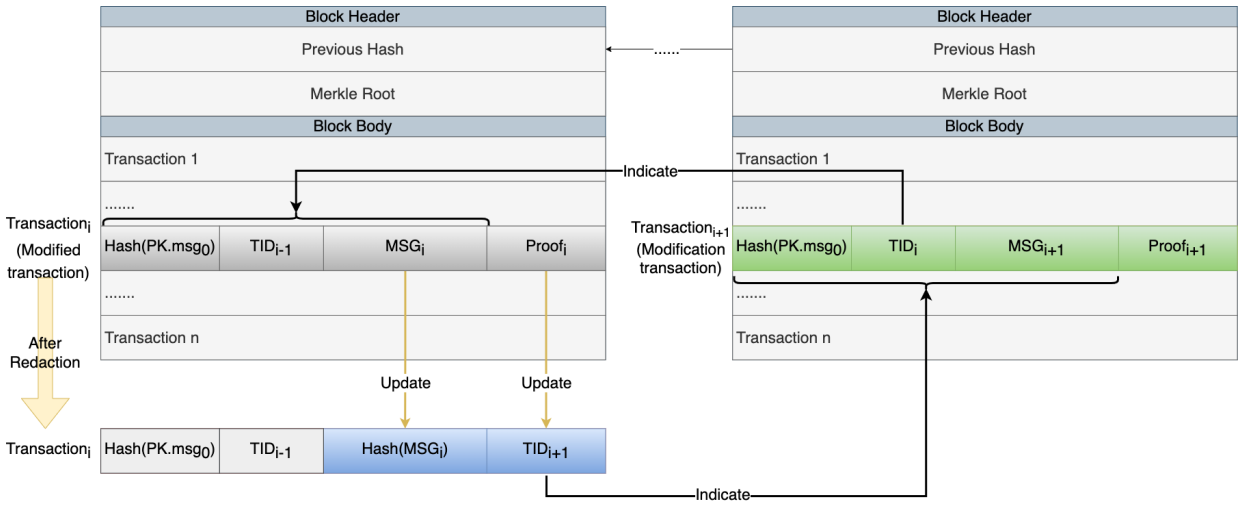

Fig. 4. Transaction modification

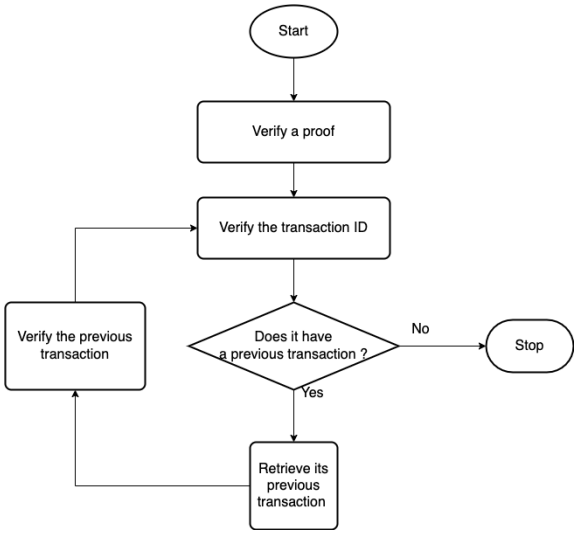

Fig. 5. Verifying modification history

D. Reversion attack

Meta-transaction-based redaction solutions enable redaction through mining modification transactions by miners. Additionally, the modification history remains in blocks, preventing adversaries from simply reverting a redacted block to its previous state [46]. Consequently, in terms of the finality, the fault tolerance to reversion attacks relies solely on consensus schemes of blockchain systems [47].

E. Double spending attack

Double spending is one of the most common risks in blockchain where a cryptocurrency can be spent more than once, leading to fraudulent transactions [48]–[50]. Redaction, which involves removing traceable transaction history from the ledger, can make a system more vulnerable to double-spending attacks [5]. To address this problem, we propose a redaction scheme based on meta-transaction, where miners can easily validate whether the existing transaction includes spent outputs. If the existing transaction does include spent outputs and change the amount spent currency, miners will simply reject committing the transaction to modify into a new block. Additionally, the revision history allows only the last single revision, preventing double-spending attacks. This approach ensures that potentially double spending transactions are not included in the blockchain, maintaining the integrity and security of the system.

F. Applications

This section examines various applications where this approach can be effectively employed, demonstrating its adaptability and usability in diverse scenarios. Notably, the solution is designed in a decentralised manner without relying on central authorities or special components, making it applicable to both permissioned and permissionless blockchains. Transaction owners possess the permission to modify their own transactions, while simultaneously having the responsibility of managing the mast key.

As immutability is one of the fundamental properties of blockchain that ensures data integrity, redaction can pose risks to integrity, particularly in cases such as the double spending problem in cryptocurrency. However, the proposed solution based on the meta-transaction scheme can provide auditability by disallowing updates to spent currency. Although an additional verification feature is required for UTXO management to check for removing or modifying transactions with UTXO, this solution can be adapted for cryptocurrency blockchains. Immutable blockchains often face challenges related to harmful or illegal content. For example, critical and illegal content, such as leaked private keys, illegal prime numbers for DVDs or links to indecent services can be found on Bitcoin [51]. This solution can effectively address these problems by removing such content without altering asset transfers, thereby maintaining blockchain integrity.

Smart contracts can also expose critical security vulnerabilities that can be potentially harmful to their users. Smart contract is a software program running on top of the virtual machine of blockchain, adapting to many cryptocurrency blockchains to manage assets. However, attackers have specifically targeted the vulnerabilities of smart contracts, leading to significant losses of assets in blockchain systems, such as a decentralised autonomous organisation (DAO) contract bug or parity wallet incidents [52], [53]. By employing the proposed solution, smart contract distributors can fix their smart contracts and update them without resorting to expensive solutions like hard forks. This approach offers an efficient way to enhance the security and reliability of smart contracts, mitigating potential risks and protecting users from potential attacks.

V. Experiments and Evaluation

TABLE II
Simulation Environments

Items	Comments
CPU	Intel Core i7-10610U
Operating system	Ubuntu
Transaction	Bitcoin
# Transactions	16,324 [1]
Library	ZoKrates for zk-SNARK [2] ZoKrates pyCrypto [3]
zk-SNARK	GROTH 16 [54] BN128 [55]

We implement a simulator written in Python and conduct performance measurements on Intel Core i7-10610U CPU. The evaluation is carried out using 16,324 real Bitcoin transactions which are downloaded and utilised within the simulator. The details of the simulation environment are presented in Table II.

TABLE III
Performance

Item	Value	Comments
Proof generation	10.55 sec	By transaction owners
Verification	7.8 msec	By miners or storage nodes
Proof size	320 Bytes	The size of proof for zk-SNARK

The average proof generation time performed by transaction owners is 10.55 seconds, while the average verification time performed by miners or storage nodes is 7.8 milliseconds as shown in Table III. The size of proof is 320 Bytes, which is the main factor contributing to the increase in transaction size for redaction. The verification performance is scalable, not affecting system performance, while the overhead of transaction size can have a negative impact on storage efficiency for blockchain systems with small transaction size such as Bitcoin. However, this redaction solution provides a transaction removal feature, which can reduce the storage requirement for data-centric blockchains that have a large size of transactions.

The variation in verification time according to the number of modifications is simulated to analyse its impact. We implement a transaction storage system utilising files. Given the previously highlighted fast verification time of the proposed solution, fluctuations in verification time is expected to be primarily driven by file retrieval performance. Figure 6 shows the increment in the verification time based on the number of modifications. It is evident that the verification time exhibits almost linear growth in spite of the fluctuations. When the length of the chain increases from 1 to 100, the verification time experiences a 147% increase, rising from 7.8 msec to 11.5 msec. While the decline in performance is not substantial, frequent modifications can compromise the performance of the blockchain system. Consequently, we encourage users to maintain their modification history short through the imposition of redaction fees.

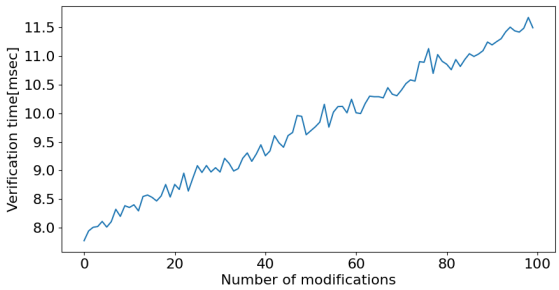

Fig. 6. Verification time according to the number of modifications

VI. Conclusion

We present a novel approach to redacting blockchain data in a decentralised manner. Our solution leverages a one-time cryptographic key generation scheme and zk-SNARK to enhance anonymity within a chained architecture designed to manage modification history based on meta-transaction. This approach effectively prevents identity tracing, a threat to privacy, by employing a one-time cryptographic key generation scheme and utilising zk-SNARK to hide the cryptographic keys. The deterministic key generation scheme empowers users to easily manage their cryptographic keys, mitigating key exposure issues potentially caused by central authorities. Furthermore, our solution addresses the problem of private key leakage by concealing signatures to verify redaction permission through zk-SNARK. By implementing a differential rate of redaction fees, our approach minimises transaction verification time, encouraging users to keep their modification history short.

[1] https://blockchair.com/bitcoin
[2] https://zokrates.github.io/
[3] https://github.com/Zokrates/pycrypto

References

[1] S. R. Niya, J. Willems, and B. Stiller, "On-chain iot data modification in blockchains," *arXiv preprint arXiv:2103.10756*, 2021.
[2] A. B. Haque, A. N. Islam, S. Hyrynsalmi, B. Naqvi, and K. Smolander, "Gdpr compliant blockchains–a systematic literature review," *IEEE Access*, vol. 9, pp. 50 593–50 606, 2021.

[3] U. Tatar, Y. Gokce, and B. Nussbaum, "Law versus technology: Blockchain, gdpr, and tough tradeoffs," *Computer Law & Security Review*, vol. 38, p. 105454, 2020.

[4] S. Schwerin, "Blockchain and privacy protection in the case of the european general data protection regulation (gdpr): a delphi study," *The Journal of the British Blockchain Association*, vol. 1, no. 1, 2018.

[5] D. Zhang, J. Le, X. Lei, T. Xiang, and X. Liao, "Exploring the redaction mechanisms of mutable blockchains: A comprehensive survey," *International Journal of Intelligent Systems*, vol. 36, no. 9, pp. 5051–5084, 2021.

[6] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha, and X. Du, "Scalable and redactable blockchain with update and anonymity," *Information Sciences*, vol. 546, pp. 25–41, 2021.

[7] M. Schellekens, "Does regulation of illegal content need reconsideration in light of blockchains?" *International journal of law and information technology*, vol. 27, no. 3, pp. 292–305, 2019.

[8] T. Ye, M. Luo, Y. Yang, K.-K. R. Choo, and D. He, "A survey on redactable blockchain: Challenges and opportunities," *IEEE Transactions on Network Science and Engineering*, 2023.

[9] M. Florian, S. Henningsen, S. Beaucamp, and B. Scheuermann, "Erasing data from blockchain nodes," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 367–376.

[10] 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)*. IEEE, 2018, pp. 364–370.

[11] 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," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 420–438.

[12] blockchair, ""blockchain explorer, analytics and web services"," 2022. [Online]. Available: "https://blockchair.com/bitcoin"

[13] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain–or–rewriting history in bitcoin and friends," in *2017 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2017, pp. 111–126.

[14] L. Cheng, J. Liu, C. Su, K. Liang, G. Xu, and W. Wang, "Polynomial-based modifiable blockchain structure for removing fraud transactions," *Future generation computer systems*, vol. 99, pp. 154–163, 2019.

[15] D. Grigoriev and V. Shpilrain, "Rsa and redactable blockchains," *International Journal of Computer Mathematics: Computer Systems Theory*, vol. 6, no. 1, pp. 1–6, 2021.

[16] A. Dorri, S. S. Kanhere, and R. Jurdak, "Mof-bc: A memory optimized and flexible blockchain for large scale networks," *Future Generation Computer Systems*, vol. 92, pp. 357–373, 2019.

[17] J. Shen, X. Chen, Z. Liu, and W. Susilo, "Verifiable and redactable blockchains with fully editing operations," *IEEE Transactions on Information Forensics and Security*, 2023.

[18] I. Puddu, A. Dmitrienko, and S. Capkun, "μchain: How to forget without hard forks," *Cryptology ePrint Archive*, 2017.

[19] S. M. Abd Ali, M. N. Yusoff, and H. F. Hasan, "Redactable blockchain: comprehensive review, mechanisms, challenges, open issues and future research directions," *Future Internet*, vol. 15, no. 1, p. 35, 2023.

[20] E. Politou, F. Casino, E. Alepis, and C. Patsakis, "Blockchain mutability: Challenges and proposed solutions," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1972–1986, 2019.

[21] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for c: Verifying program executions succinctly and in zero knowledge," in *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Springer, 2013, pp. 90–108.

[22] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Scalable zero knowledge via cycles of elliptic curves," *Algorithmica*, vol. 79, pp. 1102–1160, 2017.

[23] S. Josefsson and I. Liusvaara, "Edwards-curve digital signature algorithm (eddsa)," Tech. Rep., 2017.

[24] S. Su, F. Yuan, Y. Yuan, L. Zeng, and C. Chen, "Vofsq: an efficient file-sharing interactive verification protocol," in *2021 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2021, pp. 1–7.

[25] A. Garoffolo, D. Kaidalov, and R. Oliynykov, "Zendoo: A zk-snark verifiable cross-chain transfer protocol enabling decoupled and decentralized sidechains," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 1257–1262.

[26] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 459–474.

[27] S. Bowe, A. Gabizon, and M. D. Green, "A multi-party protocol for constructing the public parameters of the pinocchio zk-snark," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 64–77.

[28] M. Veeningen, "Pinocchio-based adaptive zk-snarks and secure/correct adaptive function evaluation," in *International Conference on Cryptology in Africa*. Springer, 2017, pp. 21–39.

[29] S. Setty, "Spartan: Efficient and general-purpose zksnarks without trusted setup," in *Annual International Cryptology Conference*. Springer, 2020, pp. 704–737.

[30] 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*. Springer, 2017, pp. 152–182.

[31] S. Nath, "Energy efficient sensor data logging with amnesic flash storage," in *2009 International Conference on Information Processing in Sensor Networks*. IEEE, 2009, pp. 157–168.

[32] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, pp. 36–63, 2001.

[33] M. Niitsoo, "Deterministic random oracles," in *International Conference on Provable Security*. Springer, 2012, pp. 88–103.

[34] B. Kieu-Do-Nguyen, C. Pham-Quoc, N.-T. Tran, C.-K. Pham, and T.-T. Hoang, "Low-cost area-efficient fpga-based multi-functional ecdsa/eddsa," *Cryptography*, vol. 6, no. 2, p. 25, 2022.

[35] W. Cao, H. Shi, H. Chen, J. Chen, L. Fan, and W. Wu, "Lattice-based fault attacks on deterministic signature schemes of ecdsa and eddsa," in *Cryptographers' Track at the RSA Conference*. Springer, 2022, pp. 169–195.

[36] A. M. Pinto, "An introduction to the use of zk-snarks in blockchains," in *Mathematical Research for Blockchain Economy: 1st International Conference MARBLE 2019, Santorini, Greece*. Springer, 2020, pp. 233–249.

[37] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *Open Problems in Network Security: IFIP WG 11.4 International Workshop, iNetSec 2015, Zurich, Switzerland, October 29, 2015, Revised Selected Papers*. Springer, 2016, pp. 112–125.

[38] Y. Zhu, R. Guo, G. Gan, and W.-T. Tsai, "Interactive incontestable signature for transactions confirmation in bitcoin blockchain," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2016, pp. 443–448.

[39] K. Li, H. Li, H. Hou, K. Li, and Y. Chen, "Proof of vote: A high-performance consensus protocol based on vote mechanism & consortium blockchain," in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2017, pp. 466–473.

[40] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "Blockchain: A distributed solution to automotive security and privacy," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 119–125, 2017.

[41] Y. Luo, "Difference between ecdsa with ed25519 and their future," *Highlights in Science, Engineering and Technology*, vol. 39, pp. 1122–1126, 2023.

[42] L. Zhao, L. Zhong, and J. Zhang, "Traceable one-time address solution to the interactive blockchain for digital museum assets," *Information Sciences*, vol. 625, pp. 157–174, 2023.

[43] R. Addas and N. Zhang, "Formal security analysis and performance evaluation of the linkable anonymous access protocol," in *Information and Communication Technology: Second IFIP TC5/8 International Conference, ICT-EurAsia 2014, Bali, Indonesia, April 14-17, 2014. Proceedings 2*. Springer, 2014, pp. 500–510.

[44] Z. Wang, H. Yu, Z. Zhang, J. Piao, and J. Liu, "Ecdsa weak randomness in bitcoin," *Future Generation Computer Systems*, vol. 102, pp. 507–513, 2020.

[45] S.-G. Liu, W.-Q. Chen, and J.-L. Liu, "An efficient double parameter elliptic curve digital signature algorithm for blockchain," *IEEE Access*, vol. 9, pp. 77 058–77 066, 2021.

[46] M. S. Dousti and A. Küpçü, "Moderated redactable blockchains: A definitional framework with an efficient construct," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2020 International Workshops, DPM 2020 and CBT 2020, Guildford, UK, September 17–18, 2020, Revised Selected Papers 15*. Springer, 2020, pp. 355–373.

[47] S. Zhang and J.-H. Lee, "Analysis of the main consensus protocols of blockchain," *ICT express*, vol. 6, no. 2, pp. 93–97, 2020.

[48] A. Begum, A. Tareq, M. Sultana, M. Sohel, T. Rahman, and A. Sarwar, "Blockchain attacks analysis and a model to solve double spending attack," *International Journal of Machine Learning and Computing*, vol. 10, no. 2, pp. 352–357, 2020.

[49] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *2017 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE, 2017, pp. 2567–2572.

[50] K. Huang, Y. Mu, F. Rezaeibagha, X. Zhang, and T. Chen, "Building blockchains with secure and practical public-key cryptographic algorithms: Background, motivations and example," *IEEE Network*, vol. 35, no. 6, pp. 240–246, 2021.

[51] R. Matzutt, O. Hohlfeld, M. Henze, R. Rawiel, J. H. Ziegeldorf, and K. Wehrle, "Poster: I don't want that content! on the risks of exploiting bitcoin's blockchain as a content store," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 1769–1771.

[52] B. Jiang, Y. Liu, and W. K. Chan, "Contractfuzzer: Fuzzing smart contracts for vulnerability detection," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 259–269.

[53] D. He, Z. Deng, Y. Zhang, S. Chan, Y. Cheng, and N. Guizani, "Smart contract vulnerability analysis and security audit," *IEEE Network*, vol. 34, no. 5, pp. 276–282, 2020.

[54] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*. Springer, 2016, pp. 305–326.

[55] C. Reitwiessner, "Eip-196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128," *Ethereum Improvement Proposals*, no. 196, 2017. [Online]. Available: "https://eips.ethereum.org/EIPS/eip-196"