POLITECNICO
MILANO 1863

Blockchain and Distributed Ledger Technologies:

Principles, Applications and Research Challenges

Zero Knowledge Proof in blockchain

Author:
Sofia Martellozzo
10623060

September, 2023

Contents

1 Introduction

Blockchain technology, which has recently garnered substantial attention, is commonly recognized as a public, decentralized, and distributed ledger system. Within the blockchain environment, every historical transaction is meticulously recorded and securely stored. However, the very attributes that make blockchain revolutionary – its openness and transparency – also introduce vulnerabilities, particularly concerning the privacy of transaction data. Malicious actors may exploit these characteristics to illicitly access sensitive information, such as transaction amounts, account addresses, and account balances.

To address this pressing concern, cryptographic innovation emerges as a formidable ally in the blockchain landscape. Zero-knowledge proof (ZKP), a cryptographic technique, plays a pivotal role in verifying transactions within the blockchain without compromising the confidentiality of sensitive data. ZKP enables the verification of whether a transaction prover possesses sufficient funds without exposing any private transaction details.

Transactions within blockchain systems inherently comprise individual privacy elements, including transaction amounts, account addresses, and account balances. However, the very openness and transparency of blockchain expose these details to anyone who seeks them, leading to a plethora of security and privacy challenges.

Public ledger verification, upon which blockchain transactions are based, amplifies these concerns. The process accumulates extensive data on the blockchain, leading to inefficiencies in verifying the entire transaction chain. To remedy these inefficiencies and enhance privacy protection, a novel solution emerges – an efficient block verification mechanism rooted in zero-knowledge proof (ZKP). This innovative mechanism not only fortifies user privacy but also expedites data block verification. Notably, it eliminates the need to place the entire transaction on the blockchain for verification. Instead, ZKP generates verification proofs and root hashes using transaction information, storing them within smart contracts for validation. Once validated, the system seamlessly incorporates the complete transaction, updates the transaction status in a cloud database, and securely integrates it into the blockchain. In this manner, ZKP strengthens the blockchain's privacy protection capabilities, while smart contracts streamline block verification, saving valuable time.

Blockchain, as an underlying technology, operates as a decentralized and distributed database. Within this framework, valuable information is forever etched into data, forming the cornerstone of blockchain: blocks. Technically, a block serves as a data structure that not only records transactions but also traces the flow of currency in commerce. It employs cryptographic techniques to ensure the immutability and inviolability of these records. The essence of blockchain lies in its fundamental principles: decentralization, openness, autonomy, and anonymity. This decentralization grants blockchain the capacity to achieve scalability, robustness, privacy, and load balancing, all while sidestepping the vulnerabilities associated with centralized structures and single points of failure.

However, despite the early success in addressing security and privacy challenges, blockchain technology is not without its share of security concerns. In this context, zero-knowledge proof (ZKP) solutions emerge as a potent means to tackle issues such as transaction trust, privacy protection, data encryption, and interaction within the blockchain ecosystem.

2 Zero Knowledge Proof

Zero Knowledge Proof (ZKP) represents a transformative breakthrough in the world of cryptography and data privacy. As their name suggests, ZKP are cryptographic protocols that do not disclose the data or secrecy to any eavesdropper during the protocol.
Zero-knowledge proofs first appeared in a 1985 paper [1] which provides a definition of zero-knowledge proofs:

" A zero-knowledge protocol is a method by which one party (the prover) can prove to another party (the verifier) that something is true, without revealing any information apart from the fact that this specific statement is true. "

Within this protocol, the verifier gains conviction that the prover possesses secret data without compromising the confidentiality of sensitive information, such as the provider's data and identity, and the verifier's own identity.

Zero-knowledge proofs solve the privacy problem by eliminating the need to reveal information to prove validity of claims. The zero-knowledge protocol uses the statement (called a ‘witness’) as input to generate a succinct proof of its validity. This proof provides strong guarantees that a statement is true without exposing the information used in creating it.

A zero-knowledge proof allows you to prove the truth of a statement without sharing the statement’s contents or revealing how you discovered the truth. To make this possible, zero-knowledge protocols rely on algorithms that take some data as input and return ‘true’ or ‘false’ as output. A zero-knowledge protocol must satisfy the following criteria:

1. Completeness: If the input is valid, the zero-knowledge protocol always returns ‘true’. Hence, if the underlying statement is true, and the prover and verifier act honestly, the proof can be accepted.

2. Soundness: If the input is invalid, it is theoretically impossible to fool the zero-knowledge protocol to return ‘true’. Hence, a lying prover cannot trick an honest verifier into believing an invalid statement is valid (except with a tiny margin of probability).

3. Zero-Knowledge: The verifier learns nothing about a statement beyond its validity or falsity (they have “zero knowledge” of the statement). This requirement also prevents the verifier from deriving the original input (the statement’s contents) from the proof.

The ZKP framework operates with two pivotal entities: the prover and the verifier. Implementation encompasses three distinct phases [7] :

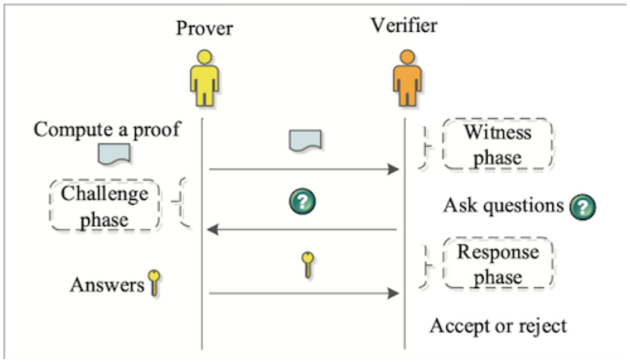

Figure 1: ZKP framework

- Witness: With a zero-knowledge proof, the prover wants to prove knowledge of some hidden information. The secret information is the "witness" to the proof, and the prover's assumed knowledge of the witness establishes a set of questions that can only be answered by a party with knowledge of the information. Thus, the prover starts the proving process by randomly choosing a question, calculating the answer, and sending it to the verifier. The prover computes a proof that contains its statement. Then, the proof is transmitted to the verifier.

- Challenge: The verifier randomly picks another question from the set and asks the prover to answer it. The verifier asks the prover several questions.

- Response: The prover accepts the question, calculates the answer, and returns it to the verifier. The prover's response allows the verifier to check if the former really has access to the witness. To ensure the prover isn't guessing blindly and getting the correct answers by chance, the verifier picks more questions to ask. By repeating this interaction many times, the possibility of the prover faking knowledge of the witness drops significantly until the verifier is satisfied. The prover answers these questions, which can be used for the verifier to accept or reject the generated proof.

In the above phases, no private data will be leaked. Furthermore, this framework has the ZKP properties required: it is Complete, Sound and Zero-Knowledge.

ZKP protocols can be of two types:

- Interactive, requiring the prover and verifier to send multiple rounds of messages back and forth.

- Non-interactive, requiring the prover to send only one letter to the verifier according to the protocol.

In some protocols, the size of the proof is related to the size of the problem, and the more complex the situation, the lower the guarantee. In other protocols, the proof size is the same regardless of the complexity of the problem. The Probabilistic Checkable Proofs (PCP) theorem states that NP assertions have probabilistic proofs verified in average proof size and logarithmic polynomial time. A universal, non-interactive, constant size ZKP protocol has been the goal of cryptography researchers for many years.

While revolutionary, interactive proving had limited usefulness since it required the two parties to be available and interact repeatedly. Even if a verifier was convinced of a prover's honesty, the proof would be unavailable for independent verification (computing a new proof required a new set of messages between the prover and verifier). To solve this problem, non-interactive zero-knowledge proof was suggested, where the prover and verifier have a shared key. This allows the prover to demonstrate their knowledge of some information (i.e., witness) without providing the information itself. Unlike interactive proofs, noninteractive proofs required only one round of communication between participants (prover and verifier). The prover passes the secret information to a special algorithm to compute a zero-knowledge proof. This proof is sent to the verifier, who checks that the prover knows the secret information using another algorithm. Non-interactive proving reduces communication between prover and verifier, making ZK-proofs more efficient. Moreover, once a proof is generated, it is available for anyone else (with access to the shared key and verification algorithm) to verify.

ZKP's versatility is demonstrated through its applicability in diverse domains, including anonymous verifiable voting, secure digital asset exchange, remote biometric authentication, and secure auctions.

The perfect combination of ZKP and blockchain can provide an excellent solution to the current dilemma of blockchain. On the one hand, the nature of blockchain is open and transparent, which makes it limited in terms of privacy and data security. On the other hand, how to solve the problem of algorithm performance, such as improving throughput and response speed are the biggest problems facing the large-scale implementation of blockchain.

Significantly, ZKP finds a profound application within the blockchain environment. It allows verifiers to confirm a prover's possession of an adequate transaction amount without revealing any private transaction data. The verification process unfolds in several steps:

1. Proof Generation: ZKP initiates the process by generating a proof that invariably contains the prover's claim regarding their transaction amount.

2. Transmission to Verifier: The generated proof is then transmitted to the verifier for evaluation.

3. Computation and Conclusion: Upon receipt, the verifier conducts predefined computations on the proof, culminating in a final computation result.

4. Claim Acceptance: Based on this result, the verifier reaches a conclusion regarding the prover's claim. If accepted, it signifies that the prover indeed possesses the required transaction amount. This entire verification process can be securely recorded on the blockchain, immune to any potential falsification attempts.

2.1 Example to better understand

To better understand the concept in this chapter the 'Two balls and the colour-blinf friend' example is illustrated, as an example of how ZKP works in real life applications [2].

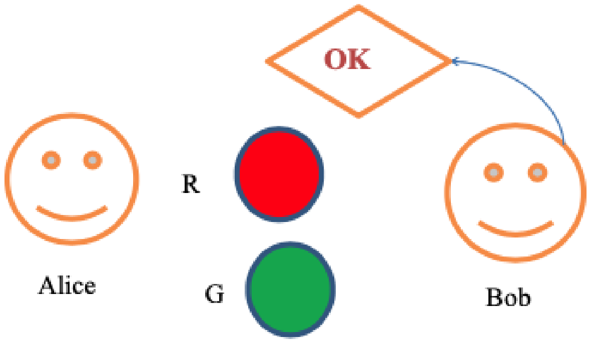

Figure 2: Two balls and the colour-blind friend

Imagine two friends: Bob, a color-blind person (he can't distinguish between Red (R) and Green (G)), and Alice, that is not color-blind. Alice has two balls, R and G, but otherwise identical. Them are identical and it's skeptical for Alice's friend Bob that they can be distinguished.
Alice want to prove to Bob that they are, in fact, different-colored, but nothing else; in particular, Alice don't want to disclose which one is the R and which one is the G ball.

To do so, Alice gives the balls to Bob, and Bob is putting them behind his back. Next, Bob takes and shows one of the balls from behind his back. Then he puts it back behind his back again and then chooses to disclose just one of the two balls, to pick one of the two at random with the same probability. He is going to tell you, "Have I changed the ball?" All this is then repeated as often as required. By looking at their colors, of course, you can tell with certainty if he changed them or not. On the other hand, if they were the same color and therefore indistinguishable, with a probability higher than 50%, there is no way you could guess correctly.

Since you have chances of randomly recognizing a different switch / non-switch is 50%, the chances of random success are zero ("soundness") at each switch / non-switch.
If Alice repeat this "proof" several times (e.g. 10 times) with Bob, he should be persuaded ("completeness") that the balls are actually colored differently. The aforementioned proof is "zero-knowledge" because Bob will never learns

which ball is R or G ; indeed, he does not gain knowledge of how to differentiate the balls.

In order to better understand the computations behind ZKP and its properties, we show another example in which ZKP is used to verify whether the prover owns some private information. This example can be applied for the secure authentication in the environment of blockchain. In this example, we suppose that the prover owns the secret number s. The implementation of this example includes three steps, which are described as follows:

1. The prover first computes $v = s^2 modn$ where $n = pq$, p and q are two large private primes.
$$\sqrt{n} \leq s \leq n - 1$$
s is not equal to p or q. Based on the reasonable setup of p and q, the adversary cannot extract the private s from v. In addition, the prover selects a random integer r, where $1 \leq r \leq n - 1$. The prover computes $x = r^2 modn$. x is sent to the verifier.

2. Next the verifier selects a random bit $\alpha \in 0, 1$. α is sent to the prover.

3. Then, if $\alpha = 0$ the prover sets the proof $\gamma = r$.If $\alpha = 1$, the prover calculates $\gamma = rsmodn$. The prover sends γ to the verifier. Finally, the verifier verifies γ. If $\gamma^2 = x \cdot v^\alpha modn = r^2 \cdot s^{2\alpha} modn$, the verifier accepts $gamma$.

This example satisfies properties of completeness, soundness, and zero-knowledge, which are described as follows.

Completeness can insure that the prover can provide correct $\gamma = r$ or $\gamma = rsmodn$ to the verifier. Then, the verifier can verify the received γ and accept it.

If the prover does not own the secret number s, soundness can guarantee that the verifier will reject the received γ with the probability 1/2 in each verification process. If γ is verified t times, the probability that the verifier can be fooled is $(1/2)^t$.

Zero-knowledge means that the verifier only knows v, x, and γ in each verification process. The verifier cannot obtain the prover's secret s.

2.2 z-STARK and z-SNARK

There are several ZKP models.

2.2.1 zkSNARK

Zero-knowledge succinct non-interactive argument of knowledge (zkSNARK) is an improved ZKP mechanism with the following qualities:

- Zero-knowledge: A verifier can validate the integrity of a statement without knowing anything else about the statement. The only knowledge the verifier has of the statement is whether it is true or false.
- Succinct: The zero-knowledge proof is smaller than the witness and can be verified quickly.
- Non-interactive: The proof is 'non-interactive' because the prover and verifier only interact once, unlike interactive proofs that require multiple rounds of communication.
- Argument: The proof satisfies the 'soundness' requirement, so cheating is extremely unlikely.
- (Of) Knowledge: The zero-knowledge proof cannot be constructed without access to the secret information (witness). It is difficult, if not impossible, for a prover who doesn't have the witness to compute a valid zero-knowledge proof.

The 'shared key' mentioned earlier refers to public parameters that the prover and verifier agree to use in generating and verifying proofs. Generating the public parameters (collectively known as the Common Reference String (CRS)) is a sensitive operation because of its importance in the protocol's security. If the entropy (randomness) used in generating the CRS gets into the hands of a dishonest prover, they can compute false proofs.

Multi-party computation (MPC) is a way of reducing the risks in generating public parameters. Multiple parties participate in a trusted setup ceremony, where each person contributes some random values to generate the CRS. As long as one honest party destroys their portion of the entropy, the ZK-SNARK protocol retains computational soundness. Trusted setups require users to trust the participants in parameter-generation. However, the development of ZK-STARKs has enabled proving protocols that work with a non-trusted setup.

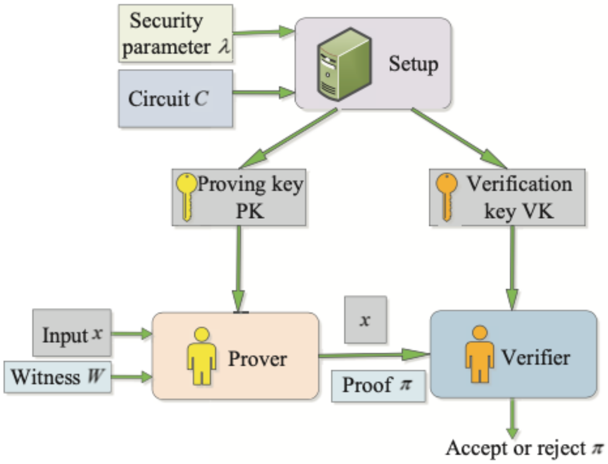

Figure 3: The framework of zkSNARK

zkSNARK mainly consists of setup, prover and verifier. Setup is a procedure that generates the proving key (PK) and the verification key (VK) by using a predefined security parameter λ and an $\mathbb{F}$-arithmetic circuit C: a circuit that all inputs are elements in a field $\mathbb{F}$, and its gates output elements in $\mathbb{F}$.

When we talk about an F-arithmetic circuit C, we mean a circuit that performs arithmetic operations using elements exclusively from a specific field F. The operations performed by this circuit, such as addition and multiplication, are governed by the rules and properties of the field F.

PK is used for generating the verifiable proof.
VK is used for verifying the generated proof.
Based on the generated PK,the input $x \in \mathbb{F}^n$ and the witness $W \in \mathbb{F}^h$, the prover generates a proof π, where C(x,W) = 0^l (the output of C is 0^l). x and W are input parameters of C. n, h, and l are dimensions of x, W, and C's output, respectively. Finally, with the usage of VK, x and π , the verifier verifies π. Accordingly to the verification result, π is accepted or rejected. Furthermore, zkSNARK has additional characteristics.

1. the verification can be executed in a short running time. In addition, the proof size is just several bytes.

2. the prover and verifier are not required to communicate with each other synchronously. The only needed proof can be verified by any verifier in a off-line way.

3. the prover algorithm can only be implemented in a polynomial time.

These SNARKs are used in the case of ZCash to verify transacyions, and they are also used in the case of Hawk to verify smart contracts. This is achieved as the privacy of users is still protected. Because zk-SNARKs can be rapidly checked, the evidence is low, the integrity of the computer can be protected without burdening non-participants.

2.2.2 zkSTARK

ZK-STARK is an acronym for Zero-Knowledge Scalable Transparent Argument of Knowledge. ZK-STARKs are similar to ZK-SNARKs, except that they are:

- Scalable: ZK-STARK is faster than ZK-SNARK at generating and verifying proofs when the size of the witness is larger. With STARK proofs, prover and verification times only slightly increase as the witness grows (SNARK prover and verifier times increase linearly with witness size).

- Transparent: ZK-STARK relies on publicly verifiable randomness to generate public parameters for proving and verification instead of a trusted setup. Thus, they are more transparent compared to ZK-SNARKs.

ZK-STARKs produce larger proofs than ZK-SNARKs meaning they generally have higher verification overheads. However, there are cases (such as proving large datasets) where ZK-STARKs may be more cost-effective than ZK-SNARKs.

2.2.3 Merkle Tree

Blockchain consists of a growing list of blocks, which are connected by using hash value. The block is a group of transac- tions, which establish a chronological sequence between them. It mainly consists of a block header and a block body.
In the block header, there is a Merkle root hash (represents a hash that can insure the integrity of all transactions in the block), a timestamp (is the current time in seconds), a random number (begins from 0 and increases for each hash calculation) and a parent block hash (is used for pointing the former block).
The block body usually stores the information about transactions.

A Merkle Tree is just a full binary tree, where each node is assigned a string:

- The leaves contain hashes of the data we'd like to commit to.
- Every internal node in the tree is assigned with a string that is a hash of its two children's string.

The string assigned to the root of the tree is referred to as the commitment. That is because even the slightest change in the underlying data causes the root to change drastically. One useful property of Merkle Trees is that it enable to prove that a certain string belongs to the underlying data, without exposing the entire data. The authentication path is the set of nodes sent to prove that a leave is in the tree, without sending the actual leaf.
It is widely believed that given the hash of some string S_0, it is infeasible to find another string $S_1 \neq S_0$ that has an identical Sha256 hash. This belief means that indeed one could not have changed the underlying data of a Merkle Tree without changing the root node's hash, and thus Merkle Trees can be used as commitment schemes.

Markle tree model is used to realize the storage and verification of user data. Most digital currencies use the tree data structure or data verification to control the time cost at the $O(\log n)$ level.
It is assumed that Alice and Bob make a transaction. The details are as follows:

1. User registration and authentication: The user must go to the authorization center to register their identity information sent to the authorization center, according to the authorization center RSA key generation algorithm RSA public key and a private key, and then the public key digital signature, the final authorization center will be digital signatures, and the private key is sent to the user, the user to save a digital signature and a private key. The digital signature must be sent to the server before the next user authentication. The transaction can start only after the signature is verified to be valid.

2. Transaction amount: The transaction between two accounts needs to realize equivalent trade according to the amount. Alice transfers a sum of money to Bob, assuming five coins, then Alice's account will lose five coins correspondingly, and Bob's account will gain five coins simultaneously. Secondly, the transaction amount should not be greater than the corresponding balance of the account. We need to prove this relationship when the transaction occurs, called scope proof.

3. Transaction address: Alice needs a plaintext address to transfer money to Bob. If the address is unknown, the transaction cannot be carried out. If the transaction address is filled in incorrectly, the funds will be transferred to another account and cannot be returned because there is no way to know who the other person is. To determine the validity of a transaction, obtain the account balance based on the address to prevent the transaction amount from exceeding the account balance.

4. The root hash of the Merkle tree:

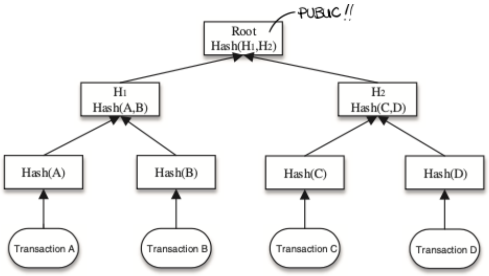

Figure 4: Merkle hash tree

according to the Merkle tree principle,the transaction occurs in Alice's account; we calculate the hash value of A and store it to the leaf node. When we want to get H_1, we must know the hash value of another transaction B. The hash value of the intermediate node $H_1 = Hash(A, B)$ is calculated again by the Hash value of transaction A and transaction B, and so on. Finally, we can get the Hash value of the root node $root = Hash(H_1, H_2$. The Hash value of the root node is public. If the user is cheating or the deal is fake, at last, the computation will change the hash value of the root, which will not be as before to hash value, thus ensuring that Alice did not dare to cheat. Otherwise, the deal will be refused because the entire network authentication, illegal trade in the validation phase will be rejected; transactions are also not packaged onto the blockchain.

5. Destruction number: The primary function of the destruction number is to prevent the problem of double spending, which simply means that the same money cannot be used twice. Alice transfers an account to Bob. If the transaction is legal, it will be packaged onto the blockchain, and the marketing will also generate a destruction number to record that the money has been consumed and cannot be consumed again. The destruction number is open to the whole network. Every legitimate transaction will have a destruction number, and it is unique. There will be multiple transactions in an account so that the destruction number will change, and only the balance corresponding to the account's latest destruction number is the account's natural balance.

6. Transaction number: Each transaction generates a number that is

bound to ZKP. ZKP only proves that the marketing is legal and does not reveal any information about the transaction. If it is a legitimate transaction, the buffer will accept the encrypted transaction. We separate the receipt of transactions from the verification of transactions. After the transaction is verified in the blockchain network, the buffer receives the encrypted transaction, but it needs to confirm that the transaction has just passed the verification. We use ZKPs to bind transactions, and buffers use binding relationships to distinguish transactions.

7. Zero-knowledge proof: After a transaction occurs in Alice's account, the transaction will contain much private information about Alice, such as the amount of Alice's transfer, the address of the transfer, the identity of the transfer, and other information; It is highly unsafe to expose these data on the blockchain. [For example, to prove that the block contains transaction A in Fig. 2, we need to construct the Merkle tree and publish the hash values of the root, B, and the intermediate node H_2. The owner of transaction A can prove that transaction A is included in the block by verifying that the generated root is consistent with the published one without knowing the actual content of other transactions.]

8. Smart contract: Advantages of using smart contract: First of all, it is decentralized. The execution of a smart contract does not rely on the participation or intervention of a third party, and computers complete the supervision and arbitration of the contract. Secondly, it cannot be tampered with and is open and transparent. Once the smart contract is deployed, all contents cannot be modified, and no party can interfere with the execution of the contract. Everything will run according to the designed code, and anyone can view it with a high degree of transparency. Last but not least, smart transparency can view contracts have lower costs than traditional contracts because they do not require supervision by a third-party intermediary and can be enforced once the contract is broken.

2.3 ZKP in blockchain

There is a need for greater privacy on the blockchain to accelerate adoption. Consumers would not be willing to receive their salaries publically or have their online purchasing history in the public for all to see. Businesses would not pay suppliers on the blockchain if their competitors could see who and how much they pay for supplies. Similarly, investment funds want to keep their strategies private and not copied before their trades have even been recorded on-chain.

A ZKP is a strong cryptographic technique, and its use in blockchain seems promising when current blockchain systems can adapt a ZKP to address particular information protection company demands or their data privacy. ZKPs can be used to ensure the validity of transactions, even though sender, recipient and other transaction data is unknown. With the consensus protocol and miners work, we can consider the blockchain as a trusted conceptual group that is trusted for accuracy and accessibility, but not trusted for privacy.

The architecture of ZKP in blockchain has two parts:

- Offchain, the prover claims that they owns enough transaction amount. The validation requester is responsible for announcing a verification task, collecting the verification result from the verifier, and paying the verification fee to the verifier.
- Onchain, the authenticity verification of the prover's claim is implemented by the verifier, which is usually a blockchain miner.

This mechanism consists of eight steps:

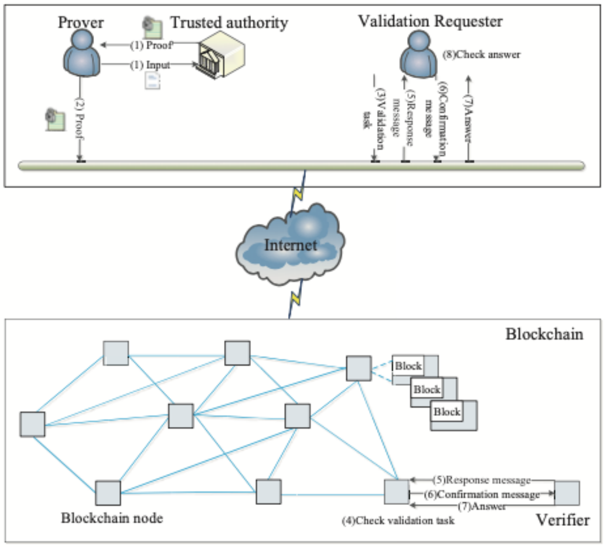

Figure 5: The architecture of ZKP in blockchain

1. Based on the the framework of zkSNARK (Section 2.2.1), trusted authority runs the setup procedure to generate the proving key and the verification key. Then the trusted authority generates a proof which always contains the prover's claim that they own enough transaction amount, rather than the transaction amount itself. The generated proof is transmitted to the prover.

2. The prover transmits the generated proof to the blockchain by the Internet. Then, the proof is stored on the blockchain. The integrity and non-tamperability of the proof can be guaranteed.

3. If the validation requester wants to know whether the prover owns enough transaction amount, the validation requester sends the verification task, which includes the task tag, the deadline for the verifier to make a response and the total amount of reward for the verification task, to the blockchain.

4. When the blockchain node receives the verification task, it checks the task tag in this task. If the task tag is valid, the verification request will be flooded to the verifier. Otherwise, the verification task will be canceled.

5. If there exists the verifier that is interested in the verification task, it will send a response message, which includes the task tag and current time, to the validation requester before the deadline.

6. When the validation requester receives the response message from the verifier, it will check this response message. If the task tag is legitimate, the reply time does not exceed the deadline, and there is no malicious behavior, the validation requester sends a confirmation message, which allows the verifier to implement the verification task, to the verifier. Otherwise, another verifier will be selected for verifying this proof.

7. The selected verifier will implement the verification task by using the verification key. After the verification of this proof, an answer, which includes the verification result, task tag, current time, and confirmation message, is transmitted back to the validation requester before the deadline.

8. When the validation requester receives the answer, it will check this answer. If the answer contains the confirmation message and the verification result is returned on time, the verification result can be accepted. Otherwise, the verification result will be discarded. Based on the accepted verification result, the validation requester can confirm that the prover owns enough transaction amount.

In addition, blockchain generates one or several new blocks, which can be used to record the process of authenticity verification without any falsification. Hence, the third party can check authenticity verification from these blocks. Furthermore, authenticity verification can be tracked by using the Merkel root in the block (Section 2.2.3).

The combination of blockchain, smart contracts, and zk-Snark algorithm can built a transaction system based on blockchain and ZKP privacy protection to achive privacy protection and efficient verification of blocks in the transaction system.

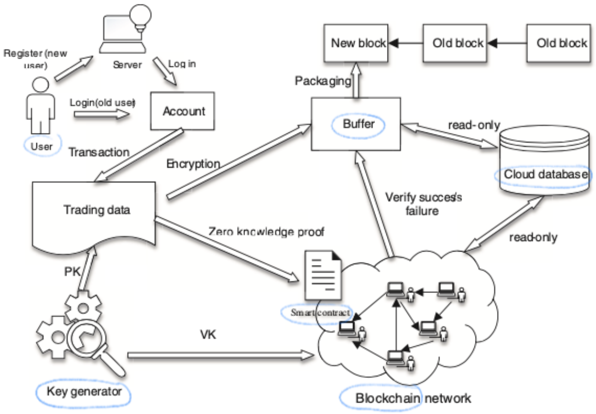

Figure 6: Data privacy protection and efficiant verification mechanism based on blockchain

The data privacy protection [8] and efficient verification scheme based on blockchain include six entities: user, key generator, smart contract, blockchain, cloud database, and buffer. Their respective functions in the system are as follows.

1. User:registers an account through the system, and then the accountancy is hashed with a random number to get an address. The address is the receiving address of each user and the transfer address used for transactions with other users.

2. Key generator: a third party that can be fully trusted. Its primary function is to generate a pair of keys P_k (proving key) and V_k (verification key), which are used by provers. The V_k is used by the verifier and is called the authentication key.

3. Smart contract: a "computer transaction agreement that implements the contract terms." It can be simply understood as a computer program, which can be automatically executed after it is set up and put on the blockchain. Unlike traditional protocols, a smart contract is immutable and transparent. When a transaction occurs in a user's account, the root hash value related to the transaction tree and ZKP needs to be written into the smart contract and uploaded to the blockchain network for verification to prevent malicious users from modifying relevant information during verification.

4. Blockchain: the core of this solution. It is a technology that realizes the characteristics of decentralization and is tamper-proof. In the blockchain network, other nodes use the authentication secret key V_k to conduct zero-knowledge verification of the whole network for the newly occurring transactions, and the transactions that pass the guarantee are packaged and connected to the chain.

5. Cloud database: records the details of each transaction. It can interact with the buffer and blockchain network, but it is read-only and cannot modify the data. After the transaction occurs, the balance of each corresponding account will change accordingly. We did not adopt the Unspent Transaction Outputs (UTXO) structure model but the account/balance model. Each legitimate transaction will be accompanied by a change in the corresponding account balance, which will be described later.

6. Buffer: all transaction information is stored in the buffer before the data is packaged and chained.To ease the pressure on blocks on the chain, only transactions that the blockchain network has verified will be packaged up by the buffer. At the same time, the cloud database updates the transaction's status and broadcasts it to the whole network. Unsuccessful transactions are discarded.

In the ZKP system, the prover accepts a proof key P_k, a private input W (also known as a witness), and a public input X. In this case, a function F exists such that $F(x, w) = 0$. The prover proved such a conclusion but did not expose private information W. Anyone can verify that F is true but can't get any information about W. The prover generates a proof π through the ZKP algorithm and sends it to the verifier. The verifier accepts the authentication key V_k, the common input X, and the proof π sent by the prover, which is computed and output ("accept" or "reject"). The privacy information exposed in the transaction process is encrypted to generate ZKP and then put into the smart contract. The verification process is completed in the smart contract.

2.3.1 Smart Contract

In the smart contract, we need to verify two problems: the first is the ownership problem. The balance of the user's account for the exchange operation must be his own to prevent malicious users from operating other people's accounts for trading. The second problem is the scope proof problem. We adopt the account balance model. When an account has a transaction, the transaction amount cannot be greater than the corresponding balance of the report. secondly, the transaction amount cannot be negative. If any of the above problems are not met, the transaction is illegal.

Suppose a transaction has been made in Alice's account, and all we need to do is prove that the transaction is legitimate. A legitimate transaction should meet two conditions at the same time. The first condition is that we need to confirm that Alice's account (all warrants) makes the transaction. Second, we need to know that the balance of Alice's account must be greater than or equal to the amount of the transaction (proof of scope). If the transfer is successful, the corresponding amount will be deducted from Alice's account, and the receiving address will be added with the corresponding amount; otherwise, the transfer will fail. After that, we can update the transaction in the cloud database and broadcast it all over the web.

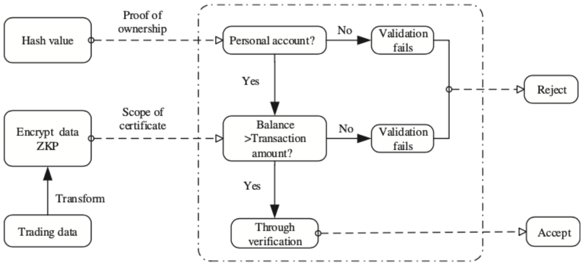

Figure 7: Smart Contract design

First, we need to prove that the money is Alice's (proof of ownership). Alice uses the private key to generate a public address and the receipt address. Then hash a generated number based on the public key and a random number. Each account will have a generated number, considered a world tree. A generated number is a leaf node that hashes with other leaf nodes to get the node's value at the next level and so on to get the value of the root hash. The root hash before and after execution is stored in the smart contract; If the root hashes are inconsistent, it can be determined that the money is not Alice's. Therefore, when a transaction occurs in Alice's account, we can judge that the money belongs to Alice according to the generation number. Meanwhile, Alice's transfer address has not been disclosed.

Secondly, we need to prove that Alice's transfer can pass the scope proof. When checking the ownership in the above steps, in the smart contract, we will check whether the balance of Alice's account is greater than or equal to the amount of transfer according to Alice's address. Here we make a judgment, and we do not know the balance of the address and the amount of transfer. Because they are encrypted we protect the sender's address and the amount.

Therefore, the primary function of ZKP is to prove whether the transaction is legal or not. The zk-snark algorithm can be used to compress the proof size and obtain an ideal time cost when verifying smart contracts. The ZKP is bound to the transaction number. If the zero-knowledge evidence is accepted, the transaction bound to the proof is also accepted. The legitimate transaction is updated in the cloud database, and the buffer packages the transaction onto the blockchain.

2.3.2 Account/Balance scheme

We use the account balance model to construct our scheme. The symbol tx represent a transaction, Z the transfer address, S the billing address, V the transfer amount, S_n the destruction number, T the transaction number and R the root hash. S_n, T and R are public. The transaction update process is shown in the following images

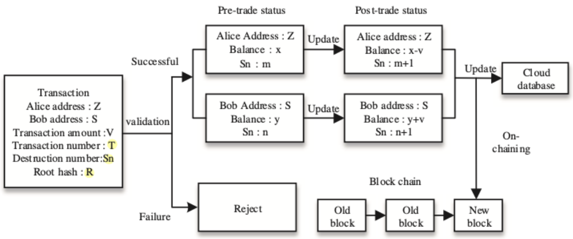

Figure 8: Account/Balance scheme design

If the validation is successful, the buffer accepts the ZKP-bound transaction, at which point the destruction number is updated, and the chain is packaged. The transaction is sent encrypted, so it is known that a legitimate transaction has been made, but not the details of the transaction, which is also updated by the cloud database.

ZKP is one of the best solutions to improve block throughput without reducing system security. A good ZKP algorithm can minimize latency as much as possible. ZKP can ensure the integrity of a remote computing process and ensure the correctness of the algorithm without disclosing private information.

The most prominent blockchain-based system that uses ZKP is ZCash, which was also the first cryptocurrency to implement zk-SNARKs.

2.4 Real applications

There are several application scenarions of ZKP.

2.4.1 Anonymous verifiable Voting

Voting is an voting essential component for guaranteeing democracy in a country or a stockholding company. Howev- er, the privacy of voters may be leaked during the process of voting. In addition, the voting result is difficult to be verified securely. ZKP is an available approach for the implementation of anonymous verifiable voting.
Based on the use of ZKP, eligible voters can cast a ballot for showing their right without leaking their identities. Besides, ZKP allows eligible voters to request a verifiable proof that their ballots are contained in the final tally by the institution that is responsible for reporting the voting result.

2.4.2 Secure exchange of digital Assets

Digital assets are a collection of binary data, which are uniquely identifiable and valuable.
If two users want to exchange their digital assets, the user's privacy, which includes identities and the content of exchanged digital assets, may be leaked

in the exchange process.
Based on the usage of ZKP, digital assets can be exchanged without leaking the user's privacy. In addition, ZKP generates a verifiable proof, which contains the process of the exchange of digital assets.

2.4.3 Secure remote biometric authentication

Remote biometric authentication is a method that can be used to identify the user's access by using their biometric modalities, such as fingerprints, facial images, iris or vascular patterns.
However, the user's biometric modalities may be leaked to an untrusted third party in the implementation of remote biometric authentication. This problem can be solved by using ZKP. In addition, ZKP generates a verifiable proof that includes the process of identifying the user's access.

2.4.4 Secure auction

Gouvernamental auction is an auction by which the government chooses the lowest bid from multiple suppliers, who sell their goods and services competitively. This auction includes two phases.
In the first phase, multiple suppliers make bids which are not known by the public.
In the second phase, these bids are opened. The gouvernament selects the winning supplier, who makes the lowest bid. However, the selection of the winning supplier may leak other losing suppliers' bids and identities.
ZKP can solve this problem. ZKP generates a verifiable proof for each losing supplier's bid. This proof verify that the difference between the losing supplier's bid and the winning supplier's bid is positive.

2.4.5 Zerocoin

In order to realize completely anonymous currency transactions in Bitcoin, Miers et al. [4] proposed an extended cryptocurrency called Zerocoin.In Zerocoin, the double spending problem is solved by ZKP, by a serial number which corresponds to funding in the commitment. In this process, transaction privacy will not be leaked. In addition, each participator tracks spent transactions according to displayed serial number. However, it still leaks the destinations and amounts of transaction.

2.4.6 Zerocash

Based on the improvement in zkSNARK, Sasson et al. [6] proposed a full-fledged ledger-based cryptocurrency called Zerocash. In Zerocash, based on the fixed address of the user, it can be paid directly and privately without interaction. Specifically, it can protect the source, destination and amount of the payment. Furthermore, Zerocash supports anonymous transactions with a variable amount.

2.4.7 Authentication systems

Authentication schemes were used for research in ZKP proofs, where a party wishes to demonstrate its identity to a second party through some secret data such as password but does not want the second party to know anything about this secret. However, a password is typically any random numbers to use in this ZKP schemes. A password proof with zero knowledge is a particular kind of certificate of knowledge that addresses the restricted size of passwords. The authentication protocols with zero know-how provide an alternative to public key cryptography authentication protocols.
Low processing and memory consumption make it particularly appropriate for use in microprocessors for smart cards that are severely restricted in processing power and storage room.

2.4.8 Ethical behavior

ONne of the uses of honest evidence or proof is to promote genuine conduct while preserving privacy within cryptographic protocols. The concept is roughly to force a user to demonstrate that his conduct is right in accordance with the Protocol using a null- knowledge evidence. Because of soundness, we know that the user must really act honestly in order to be able to provide a valid proof. Because of zero knowledge, we know that the user does not compromise the privacy of its secrets in the process of providing the proof.

There are specific "privacy coins" designed for completely anonymous transactions. Privacy-focused blockchains, such as **Zcash** and Monero, shield transaction details, including sender/receiver addresses, asset type, quantity, and the transaction timeline.

Zero-knowledge proofs are also being applied to anonymizing transactions on public blockchains. An example is **Tornado Cash**, a decentralized, non-custodial service that allows users to conduct private transactions on Ethereum. Tornado Cash uses zero-knowledge proofs to obfuscate transaction details and guarantee financial privacy. Unfortunately, because these are ”opt-in” privacy tools they are associated with illicit activity. To overcome this, privacy has to eventually become the default on public blockchains.

3 Tornado Cash

Tornado Cash [5] [9] is a smart contract-based crypto asset mixer that uses zkSNARKs to create a decentralized privacy-enhancing protocol. The code is open source and has been deployed on various blockchains, most notably Ethereum.

A common misconception among blockchain users is that pseudonymity guarantees privacy. The reality is almost the opposite. Every transaction one makes is recorded on a public ledger and reveals information about one's identity. Mixers, such as Tornado Cash, were developed to preserve privacy through "mixing" transactions with those of others in an anonymity pool, making it harder to link deposits and withdrawals from the pool.

Every transaction an address makes on a blockchain is recorded and public, revealing information about the underlying entity. As such, with graph analysis tools, one can cluster addresses together that, with reasonable confidence, possess the same owner.In response to this shortcoming, several coin mixing protocols have been proposed like Tornado Cash, to obfuscate transaction tracing, the final of which is deployed in practice.

Ethereum is the second-largest cryptocurrency platform in terms of market capitalization. However, it is the largest smart contract platform, and it is also the most used public blockchain for settling transactions. Therefore, it is imperative to understand better and assess the privacy guarantees of Ethereum quantitatively. Ethereum employs the account model. There are two types of accounts:

- EOA (Externally Owned Accounts), controlled by users via a cryptographic key-pair (a private and public key) owned by them. The private key of the EOA enables users to send transactions from that account, while the public key is used to derive an address for the EOA (the hash of the public key is its address).

- Contracts accounts, controlled by contract code. The con- tract account's address is the hash of their contract code. Accounts are referred to by their addresses (a pseudonym). Contract accounts cannot initiate transactions.

EOAs can send transactions to contract accounts that can run the code of the contract account.
Ethereum's account model has several implications from a privacy point of view. First, the account model incentivizes the reuse of accounts across many transactions. Secondly, the accounts owned by the same user can effectively be clustered.

Users can break links with their transaction history using a so-called mixer contract. Tornado Cash (TC) is the most widely used, non-custodial mixer on Ethereum. TC works as follows.

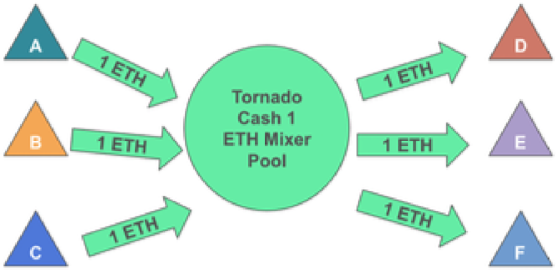

Figure 9: Tornado Cash example 1 ETH pool

Users deposit equal amounts to a TC smart contract (e.g. 1 ETH as shown above). After some time, users can withdraw their funds from the mixer contract to a freshly generated EOA by providing a zero- knowledge proof that proves that the withdrawing user is one of the depositors. Therefore, the withdrawing EOA has enhanced its privacy since it has become unlinkable to any unique depositor EOA. Note that each TC contract applies a fixed denomi- nation for the mixed funds (e.g. 1 ETH). Otherwise, linking deposits to withdraws would be trivial. A user's anonymity is defined by the number of a equal user deposits in a given pool. This is the pool's Anonymity Set.
In the example above, D's withdrawal could have come from A, B or C, so the anonymity set is 3 and the probability of correctly guessing the deposit / withdrawal connection is 1/3.
The more users that deposit in the pool, the greater the number of people that a withdrawal could have come from.
However, TC is not infallible because there are lots of ways users can compromise their deposits.

Tornado Cash is an open-source project, meaning that anyone can access and contribute to the codebase. The protocol is also permissionless, meaning that anyone can use the platform without having to go through any kind of approval process.

How can I use Tornado Cash? To use Tornado Cash, you will need a Web3 wallet (such as MetaMask) and some ETH. You can then follow these steps:

1. Go to the Tornado Cash website (https://tornado.cash/) and click on the “Mix” tab.
2. Connect your Web3 wallet to the platform by clicking on the “Connect Wallet” button.
3.Choose the amount of ETH you want to deposit and click on the “Deposit” button.
4. The platform will show you the mixing fee and the minimum deposit amount. Confirm the transaction by clicking on the “Approve” button.
5. The platform will mix your ETH with other users’ deposits. This process may take a few minutes.
6. Once the mixing is complete, you can withdraw your anonymized ETH by clicking on the “Withdraw” button.
7. Confirm the transaction by clicking on the “Submit” button.

4 Zcash

ZCash [3] can be described as an open, unauthorized, replicated ledger encrypted. It’s functioning work is almost same as Bitcoin. Transaction validators are miners and complete nodes. ZCash utilizes POW, a power supplier which provides miners with a verification of ZKP’s connected to each transaction. ZCash utilizes ZKPs to encrypt all the information and only allows approved parties to view such information by giving decryption keys.

Among the now numerous alternative cryptocurrencies derived from Bitcoin, Zcash is often touted as the one with the strongest anonymity guarantees, due to its basis in well-regarded cryptographic research.
Criminal activities may be frawn to Bitcoin due to the relatively low friction of making international payments using only pseudonyms as identifiers, but the public nature of its ledger of transactions raises the question of how much anonymity is actually being achieved. Indeed, has been demonstrated that the use of pseudonymous address in Bitcoin does not provide any meaningful level of anonymity.
In response to a growing awareness that most cryptocurrencies do not have strong anonymity guarantees, a number of alternative cryptocurrencies or other privacy-enhancing techniques have been deployed with the goal of improving on these guarantees. One of the most notable cryptocurrencies that fall into this former category is Zcash, in which users can spend shielded coins without revealing which coin they have spent through zero knowledge proof.
Zcash does not require all transac- tions to take place within the shielded pool itself: it also supports so-called transparent transactions, which are essentially the same as transactions in Bitcoin in that they reveal the pseudonymous addresses of both the senders and recipients, and the amount being sent.

Zcash (ZEC) is an alternative cryptocurrency developed as a (code) fork of Bitcoin that aims to break the link be- tween senders and recipients in a transaction.

In Bitcoin, recipients receive funds into addresses (referred to as the vOut in a transaction), and when they spend them they do so from these addresses (referred to as the vIn in a trans- action). The act of spending bitcoins thus creates a link between the sender and recipient, and these links can be followed as bitcoins continue to change hands. It is thus possible to track any given bitcoin from its creation to its current owner.

Any transaction which interacts with the so-called shielded pool in Zcash does so through the inclusion of a vJoinSplit, which specifies where the coins are com- ing from and where they are going. To receive funds, users can provide either a transparent address (t-address) or a shielded address (z-address). Coins that are held in z-addresses are said to be in the shielded pool.

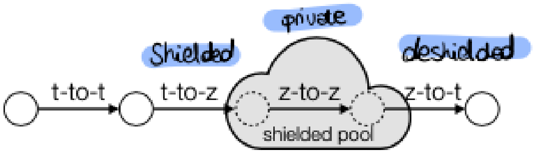

Figure 10: Illustration of different types of Zcash transactions

To specify where the funds are going, a vJoinSplit contains a list of output t-addresses with funds assigned to them (called zOut), two shielded outputs, and an encrypted memo field. The zOut can be empty, in which case the transaction is either shielded (t-to-z) or private (z-to-z), depending on the inputs. If the zOut list contains a quantity of ZEC not assigned to any address, then we still consider it to be empty (as this is simply the allocation of the miner's fee). Each shielded output contains an unknown quantity of ZEC as well as a hidden double-spending token. The shielded output can be a dummy output (i.e., it contains zero ZEC) to hide the fact that there is no shielded output. The encrypted memo field can be used to send private messages to the recipients of the shielded outputs.

To specify where the funds are coming from, a vJoin- Split also contains a list of input t-addresses (called zIn), two double-spending tokens, and a zero-knowledge proof.

The zIn can be empty, in which case the transaction is either deshielded (z-to-t) if zOut is not empty, or private (z-to-z) if it is. Each double-spending token is either a unique token belonging to some previous shielded output, or a dummy value used to hide the fact that there is no shielded input. The double-spending token does not reveal to which shielded output it belongs. The zero-knowledge proof guarantees two things.

First, it proves that the double-spending token genuinely belongs to some previous shielded output.

Second, it proves that the sum of the values in the addresses in zIn plus the values represented by the double-spending tokens is equal to the sum of the values assigned to the addresses in zOut plus the values in the shielded outputs plus the miner's fee.

ZCash emerged in 2016 when a group of scientists decided they wanted to create a cryptocurrency similar to Bitcoin but with some additional features. They developed a fork of the Bitcoin blockchain, with enhanced user security and anonymity. The scientists first invented Zerocoin, which became Zerocash not too long after its initial release. Eventually, the cryptocurrency was renamed ZCash. ZCash is a Bitcoin fork with a different hashing algorithm and security protocols, it started as Zerocash; it was later improved upon by the Electric Coin Company and rebranded ZCash. ZCash verifies ownership of coins and transactions more anonymously than Bitcoin, thus providing more security for users. Zcoin can be mined on devices and computers, but the currency is best mined on dedicated systems called application-specific integrated circuits. Zcash uses the zk-SNARK security protocol to ensure the parties involved in a transaction are verified without revealing any information to each other or the network.

Zcash is an altcoin, a category of cryptocurrency that shares many of the characteristics of Bitcoin. Many altcoins are different in their purpose and intended uses.

Zk-Snark allows for fully shielded transactions in which the sender, recipient, and amount are encrypted. This feature is a large deviation from other cryptocurrencies, where transaction transparency is an underlying concept aside from securing user information. The Bitcoin community prides itself on transparent transactions while maintaining anonymity. However, anyone interested or who has a stake in a transaction could trace the parties within it. ZCash doesn't eliminate transaction information. Instead, it encrypts it so that it cannot be tracked. The ZCash blockchain is still encrypted, but the security protocol zk-SNARK adds additional user security and anonymity. Bitcoin uses the hashing algorithm SHA-256. ZCash uses Equihash (a proof-of-work mining algorithm based on the Generalized Birthday Problem concept), which is incompatible with hardware and software designed for Bitcoin mining. It also has larger blocks and increased hashing times, which increases the network's hash rate. A cryptocurrency's hash rate is the processing power of the network of miners—it's a measure of how fast the transactions can be verified and validated to open a new block. [Hashing is the process used to convert data into a string of alphanumeric characters. The string of numbers is unique because it is created from the data in the block. Once hashed, it cannot be replicated. The hashing algorithm is the mathematical method used to create the alphanumeric string, also called the hash.]

5 Drawbacks

There are some drawbacks using zero-knowledge proofs

5.1 Hardware costs

Generating zero-knowledge proofs involves very complex calculations best performed on specialized machines. As these machines are expensive, they are often out of the reach of regular individuals. Additionally, applications that want to use zero-knowledge technology must factor in hardware costs which may increase costs for end users.

5.2 Proof verification costs

Verifying proofs also requires complex computation and increases the costs of implementing zero-knowledge technology in applications. This cost is particularly relevant in the context of proving computation. For example, ZK-rollups pay 500,000 gas to verify a single ZK-SNARK proof on Ethereum, with ZK-STARKs requiring even higher fees.

5.3 Trust assumptions

In ZK-SNARK, the Common Reference String (public parameters) is generated once and available for re-use to parties who wish to participate in the zero-knowledge protocol. Public parameters are created via a trusted setup ceremony, where participants are assumed to be honest.

But there's really no way for users to assess the honesty of participants and users have to take developers at their word. ZK-STARKs are free from trust assumptions since the randomness used in generating the string is publicly verifiable. In the meantime, researchers are working on non-trusted setups for ZK-SNARKs to increase the security of proving mechanisms.

5.4 Quantum computing threats

ZK-SNARK uses elliptic curve cryptography (ECDSA) for encryption. While the ECDSA algorithm is secure for now, the development of quantum computers could break its security model in the future.

ZK-STARK is considered immune to the threat of quantum computing, as it uses collision-resistant hashes for encryption. Unlike public-private key pairings used in elliptic curve cryptography, collision-resistant hashing is more difficult for quantum computing algorithms to break.

6 Conclusion

This project starts with an explanation to better understand the reasons that have broth so much attention around Zero Knowledge Proof. The main chapter explains what actually is this crythographic technique with two example to better understand the concept. We also went through different concept relevants in the field, such as zkSNARK, zkSTARK and Merkle Tree. The technique is then intruduced in the environment of blockchain followed by some real application of it. Other two main chapter explained two practical implementation of ZKP in blockchain: Tornado Cash (built on top of blockchain, like Ethereum) and Zcash (an actual new crypyocurrency). To finish the research I reported some drawbacks in using ZKP. For sure ZKP is a really powerfull tool still under research that could bring to relly improvements in blockchain field.

References

[1] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof systems, 17th. In *Annual ACM Symp. on Theory of Computing*, volume 10, 1985.

[2] Jahid Hasan. Overview and applications of zero knowledge proof (zkp). *Nanjing: Nanjing University of Posts and Telecommunications*, 2019.

[3] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 463–477, 2018.

[4] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE, 2013.

[5] Matthias Nadler and Fabian Schär. Tornado cash and blockchain privacy: A primer for economists and policymakers. *Available at SSRN 4352337*, 2023.

[6] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[7] Xiaoqiang Sun, F Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE network*, 35(4):198–205, 2021.

[8] Jin Wang, Wei Ou, Osama Alfarraj, Amr Tolba, Gwang-Jun Kim, and Yongjun Ren. Block verification mechanism based on zero-knowledge proof in blockchain. *Comput. Syst. Sci. Eng.*, 45(2):1805–1819, 2023.

[9] Mike Wu, Will McTighe, Kaili Wang, Istvan A Seres, Nick Bax, Manuel Puebla, Mariano Mendez, Federico Carrone, Tomás De Mattey, Herman O Demaestri, et al. Tutela: An open-source tool for assessing user-privacy on ethereum and tornado cash. *arXiv preprint arXiv:2201.06811*, 2022.