Identity Management System with Improved Blockchain Technology

(Major Project Report)

Prepared by :
Abhyuday Choumal (CS20B1001)

Project was done under Dr. Priodyuti Pradhan

Contents

1 DECLARATION **4**

2 CERTIFICATE **5**

3 Abstract **6**

4 Introduction **7**

5 Pre-requisites **10**

5.1 DIMS 10

5.1.1 Issues With Identity Today: 12

5.2 BlockChain 14

5.2.1 Features of BlockChain 16

5.2.2 Types of BlockChain: 18

5.3 Smart Contract 20

5.3.1 How do smart contracts work? 21

5.3.2 Advantages of smart contracts 24

5.3.3 Solidity: 27

5.4 Zero-knowledge proof 27

5.4.1 What Is a Zero-Knowledge Proof Blockchain? . . 28

5.4.2 Zero Knowledge Proof Algorithm: 30

5.4.3 The Benefits of Combining Zero-Knowledge Proofs With Blockchain 30

5.4.4 Zero-Knowledge Proof technology enables these powerful privacy functions that can serve a variety of use cases: 31

5.5 Zk-Snark and ZoKrates 33

5.5.1 What is a zk-SNARK? 33

5.5.2 How is a zk-SNARK Implemented? 35
5.5.3 Applying Zk-SNARKs on Ethereum: The ZoKrates Toolbox: 37
5.6 Some Genreal Question-Answers 39

6 Existing Claim Model For DIMS 41
6.1 Terminologies 41
6.2 Already existing claim identity model 42
6.3 Improved Claim identity model 45
6.4 Comparision between both Claim Identity Model 46

7 Proposed Architecture and Secheme Overview 48
7.1 Phase 1: Creation: 52
7.2 Phase 2: Transferring 54
7.2.1 Particular computation of claimPK: 54
7.2.2 The procedure of transferring: 54
7.3 Phase 3: Responding. 58
7.3.1 Particular computation of responseSK: 58
7.3.2 The procedure of responding: 59
7.4 Phase 4: Revocation. 62
7.5 Challenge-response protocol: 63

8 Security analysis: 66
8.1 The security of the bottom layer: 66
8.2 The security of the application layer: 67

9 The related works: 69

10 Conclusion 71

11 References 72

1 DECLARATION

I, **ABHYUDAY CHOUMAL** (**Roll Number: CS20B1001**), hereby declare that, this report entitled **"Identity Management System with Improved Blockchain Technology"** submitted to Indian Institute of Information Technology Raichur towards partial requirement of **Bachelor of Technology** in **Computer Science and Engineering** is an original work carried out by me under the supervision of **Dr. Priodyuti Pradhan** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold the academic ethics and honesty. Whenever an external information or statement or result is used then, that have been duly acknowledged and cited.

Raichur-584135 **Abhyuday Choumal**
May 2024

2 CERTIFICATE

This is to certify that the work contained in this project report entitled **"Identity Management System with Improved Blockchain Technology"** submitted by **Abhyuday Choumal** (**Roll Number: CS20B1001**) to the Indian Institute of Information Technology Raichur towards partial requirement of **Bachelor of Technology** in ,**Computer Science and Engineering** has been carried out by [him/her] under my supervision and that it has not been submitted elsewhere for the award of any degree.

Raichur-584135 (Dr. Priodyuti Pradhan)
May 2023 Project Supervisor

3 Abstract

The conventional centralized digital identity management system (DIMS) faces numerous vulnerabilities, including fragmented identity, a single point of failure, internal breaches, and privacy breaches. With the advent of blockchain technology, DIMSs can now be implemented within it, mitigating many issues associated with centralized third parties. However, the inherent transparency and lack of privacy in blockchain present significant challenges for DIMS. To address this, we utilize smart contracts and zero-knowledge proof (ZKP) algorithms to enhance the existing identity claim model in blockchain, aiming to achieve identity unlinkability. This enhancement effectively prevents the exposure of attribute ownership. Additionally, we develop a system prototype, which incorporates a challenge-response protocol. This protocol enables users to selectively disclose attribute ownership to service providers, safeguarding users' behavioral privacy. Through security analysis, our scheme demonstrates effective attribute privacy protection.

4 Introduction

The majority of digital identity management systems (DIMSs) operate in a centralized manner, rendering them susceptible to various forms of attacks, including single points of failure, phishing attacks, malicious tampering, and internal breaches. Incidents such as the privacy data scandal involving tens of millions of Facebook users highlight the severe consequences of centralized DIMS database breaches. Consequently, centralized DIMSs fail to ensure the confidentiality and availability of identity information.

However, many service providers (SPs) also function as identity providers (IdPs), leading to a dual role where users must register within the SP's domain to access its services. This model of isolated identities has resulted in a proliferation of digital identities on the internet, overwhelming users with numerous account credentials. While federated and user-centric identity models have been introduced to address this issue by enabling multiple SPs to share the same IdP, users still face challenges. Authentication processes for accessing services from different domains are often redirected to the IdP, compromising users' behavioral privacy.

The emergence of blockchain technology offers a promising solution to address identity-related challenges. With its tamper-proof distributed ledger technology, blockchain enables anyone to host the ledger and record transactions permanently. Leveraging this distributed ledger and the consensus mechanism between nodes, we can establish a trusted distributed digital identity management system (DIMS) in an untrusted environment. This decentralized approach facilitates the complete life cycle of identity management, including registration, authentication, authorization, and revocation, without relying on third parties. As a result, users gain control over their identities, mitigating issues such as identity fragmentation, theft, and single points of failure commonly encountered in traditional centralized DIMSs.

The claim identity model is currently receiving significant attention and is poised to replace traditional centralized digital identity management systems (DIMS). This model involves storing the mapping of user identifiers to attributes on a public and transparent distributed ledger. In this model, identity providers (IdPs) verify attribute values and issue claims on the mapping to validate the user's ownership of the attribute. However, it's crucial to note that only non-sensitive attribute information should be propagated to the distributed ledger to protect user identity privacy. Storing privacy-sensitive identity information in the blockchain is not feasible, which limits the applicability of blockchain-based DIMSs, especially in scenarios like commercial or financial industries where customers are required to provide sensitive information for KYC (Know Your Customer) procedures.

For instance, when Alice applies for a loan at a bank, the bank needs various attribute information to assess Alice's creditworthiness, such as occupation, credit history, age, income level, and asset details. This information contains sensitive identity data that should not be publicly exposed. Therefore, the key challenge lies in preventing the public disclosure of ownership of privacy attributes while ensuring that service providers (SPs) can still authenticate these privacy attributes. This issue must be addressed to make blockchain-based DIMSs viable in sensitive use cases like financial transactions.

The paper makes several key contributions:

1. Enhancement of the Claim Identity Model:The paper introduces the concept of privacy attribute tokens to enhance the claim identity model. By incorporating the attribute identifier and the user's public key in the hash preimage, the improved model prevents the exposure of ownership of privacy attributes in the public ledger.This modification achieves unlinkability between users and their identities, thereby expanding the application potential of the claim identity model.

2. Implementation of a Comprehensive Digital Identity Management System:A digital identity management system is developed to facilitate the complete life cycle management of privacy attributes.The system enables efficient and granular attribute management, laying a solid foundation for additional security mechanisms

3. Development of a Challenge-Response Protocol:The paper devises a challenge-response protocol that enables users to selectively disclose attribute ownership to service providers (SPs).Notably, during service access, authentication operations are redirected to privacy attributes via zero-knowledge proofs, bypassing the need for IdPs.This approach ensures that authentication content remains visible only to the SP, safeguarding users' behavioral privacy.

5 Pre-requisites

5.1 DIMS

Digital identities, accounts and users are unique concepts in cyber-security that have different scopes and use cases.Digital identity is the broadest of all three. It encompasses any attributes or identifiers that can be used to identify, authenticate and authorize an entity in a network. For example, API keys, digital certificates, IAM roles and service accounts.

A digital account is a type of digital identity used to access a particular resource or environment. Access to digital accounts is often protected via some sort of authentication, e.g. single sign-on (SSO), passwords or keys.
A digital user is an individual that interacts with a digital system. Interactions can include accessing a resource, performing some operations or using a service.
A single digital identity may have multiple digital accounts. For example, a project owner's identity may encompass different administrator accounts, service accounts and root profiles that offer them exclusive access to the entire infrastructure.

Digital identity (DI) can be defined as the digital representation of the information known about a specific individual or organization Such information is set of claims made by one subject about itself or another subject.

A model of identity can be seen as follows:

- Users who want to access a service
- Identity provider (IdP): is the issuer of user identity
- Service provider (SP): is the relay party imposing an identity check
- Identity (Id): a set of user's attributes
- Personal authentication device (PAD): device holding various identifiers and credentials and could be used for mobility

Identity management refers to "the process of representing, using, maintaining, deprovisioning and authenticating entities as digital identities in computer networks".

Authentication is the process of verifying claims about holding specific identities. A failure at this stage will threaten the validity in the entire system.

Identity management systems are elaborated to deal with the following core facets:

- Reducing identity theft: The problem of identity theft is becoming a major one, mainly in the online environment. The providers need a more efficient system to tackle this problem.
- Management: The number of digital identities per person will increase, so the users need convenient support to manage these identities and the corresponding authentication.
- Reachability: The management of reachability allows users to handle their contacts to prevent misuse of their addresses (spam) or unsolicited phone calls.
- Authenticity: Ensuring authenticity with authentication, integrity, and nonrepudiation mechanisms can prevent identity theft.

- Anonymity and pseudonymity: Providing anonymity prevents tracking or identifying the users of a service.
- Organizational personal data management: A quick method to create, modify, and delete work accounts are needed, especially in big organizations.

Without improved usability of identity management, for example, weak passwords set up by users on many websites, the number of successful attacks will remain high.

5.1.1 Issues With Identity Today:

One of the key enablers for today's digital economy is identity. Both businesses and users are becoming increasingly frustrated by the convoluted methods they are forced to use to interact with each other. Let us review some of the major issues.

Issue 1: Identity Sprawl and Privacy:

There isn't currently a universally accepted digital equivalent of the user's offline identity such as a passport or a driver's license. Users are issued a unique digital identity for each application they use on the Internet. This is difficult and counterproductive for users because they now have to remember all their usernames and passwords. Multiple credentials expose the user to a variety of security issues.

Federation has solved this problem to an extent by allowing the transfer of a user identity from one domain to another transparently. For the end user, it typically means that they can access online services seamlessly using an existing or valid session with an Identity Provider (IdP).

More recently, large social media companies such as Facebook have helped establish the concept of a social identity for users that can be leveraged as an alternative for some use cases. Some countries such as Estonia and Singapore are issuing digital identities so that citizens can safely identity themselves when they want to avail e-services.

Issue 2: Attribute Drift and Sync:
A digital identity is a set of claims made by one digital subject about itself or other digital subjects. For example, John Smith, an individual with an identity may have attributes such as gender, height, weight, mailing address, email address, date of birth, place of birth, citizenship, driver's license number, etc. Some of these attributes will be unique identifiers (e.g. email address, SSN, passport number, etc.) because they are uniquely associated with John's identity.

It is worth pointing out that some identifiers can be permanent for life (e.g. SSN). Some are long lived (e.g. driver's license number, passport number). But, many identifiers can be reassigned (e.g. cell phone numbers) and therefore it is possible that the same identifier is associated with another identity at different times. With the duplication of identities for every entity, both users and businesses are burdened with having to keep the attributes and identifiers in sync across the silos every time there is a change.

Issue 3: Inconsistent Security Posture:
Users have no guarantees that the identities issued are adequately secured by the institutions issuing them. Since these have tangible value, they are a juicy target for hackers and can result in identity theft (see below).

Issue 4: Identity Theft:
In the offline world, identity documents are issued by trusted entities that design and keep updating them in a manner to deter forgery and counterfeiting. For example, your driver license is strongly linked only to you (e.g. with a picture, a fingerprint and a number of anti-forgery mechanisms visibly embedded in

it). Therefore, if it is stolen, it is of limited value. In contrast, in the digital world, identity theft is a major concern for users. In most cases, the user or the entity have no idea that the user's digital identity has been stolen and being actively used by the fraudster.

Issue 5: Regulations:
The government holds financial institutions to high standards when it comes to "Know Your Customer" (KYC) laws. These were introduced in 2001 as part of the Patriot Act. The core idea here is that they need to know their customer (i.e. verify their identity, make sure they are real, ensure they are not on a prohibited lists and keep money laundering, terrorism financing and fraud schemes at bay).

These help establish and verify the identity of the customer by using reliable and independent data or sources of information. On the flip side, these processes can be extremely manual, cumbersome and expensive for the entities involved. It is reported that the average institution spends in the region of $60M every year ensuring adherence to these checks.

5.2 BlockChain

As a database architecture based on distributed ledger technology (DLT), blockchain was developed by Satoshi Nakamoto (2008), the unknown person behind the white paper on Bitcoin. Blockchain has the characteristics of immutability, decentralization, and disinter mediation, which allow it to be used not only for digital currency but also in other fields such as data security and supply chain traceability.

Blockchain, as the underlying technology, has recently received much attention from industry and academia. The heart of the blockchain is the recording of all transaction data into blocks and the linking of those blocks into a chain. Each block has a body with transaction data and a header with the hash value of the block before it, which is customarily called the “parent” block. This lets each block be linked vertically to be found or identified easily. Fig1 illustrates the architecture of blockchain. The blocks with timestamps are chained by hash values, which are unique and can prevent fraud because any changes in the block will immediately cause a change in the hash value. The three fundamental mechanisms of blockchain that make it unique are the distributed ledger, encryption, and consensus mechanism.

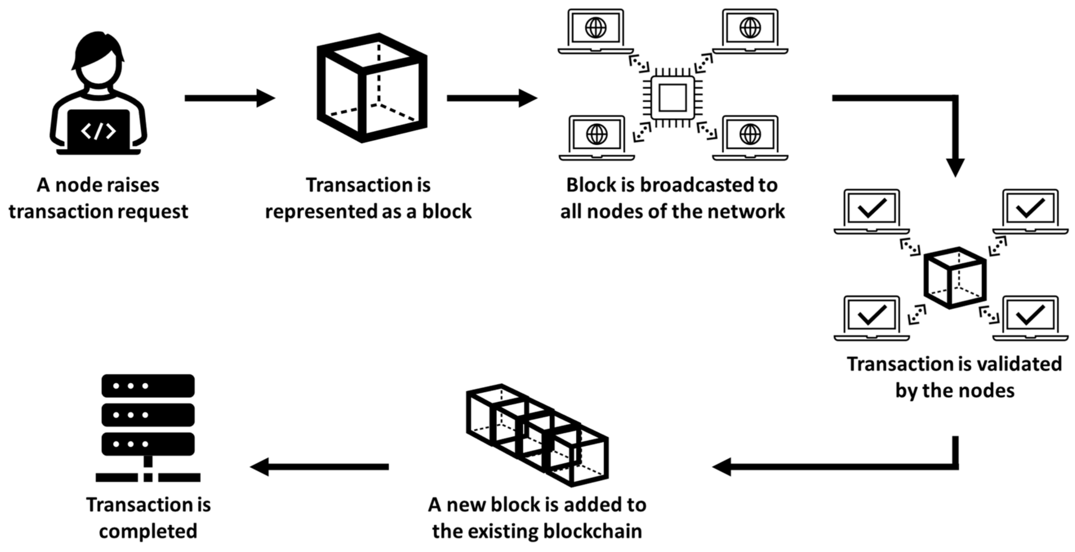
A node raises transaction request
Transaction is represented as a block
Block is broadcasted to all nodes of the network
Transaction is validated by the nodes
A new block is added to the existing blockchain
Transaction is completed

Figure 5.1: BlockChain Transaction steps.

A blockchain-based database can be considered a distributed system with an implementation layer that offers protection for data integrity because, in this distributed system, all the transactions will be consistently recorded in the ledgers of all the participants. Rather than relying on a central server to store and validate data, each node in this network has a duplicate of the ledger, which can be updated independently.

To prevent unauthorized changes, the transaction data will be encrypted using various algorithms and signed with a digital signature indicated that the Elliptic Curve Cryptography (ECC) algorithm and the Rivest-Shamir-Adleman (RSA) algorithm are the two most common encryption algorithms in blockchain, both of which belong to asymmetric encryption. ECC has become more popular in recent years, probably because of its outstanding performance, such as its lower time complexity and smaller key size, which means less capacity is needed and less energy is consumed. The benefits of ECC make it popular enough to be adopted by multiple popular protocols such as Bitcoin, secure shell (SSH), and Ethereum.

5.2.1 Features of BlockChain

There are many features of Blockchain. Following are the main features.

Decentralized :

The network is decentralized meaning a group of nodes maintains the network making it decentralized. This is one of the key features of blockchain technology. Blockchain puts us users in a straightforward position. As the system doesn't require any governing authority, we can directly access it from the web and store our assets there.

Immutability :

Immutability is something that can't be changed or altered. This one is the top Blockchain features that ensures that the technology will remain as it is a permanent, unalterable network. As it is distributed system, every node on the system has a copy of the digital ledger. When a transaction is added every node needs to check its validity. If the majority thinks it's valid, then it's added to the ledger. This promotes transparency and makes it corruption-proof. Without the majority consent from the nodes, no one can add any transaction blocks to the ledger.Once the transaction blocks added to the ledger, no one can change it. Thus, any user on the network won't be able to edit, delete or update it.

Enhanced Security :

As it gets rid of the need for a central authority, no one can just simply change any characteristics of the network for their benefit. Using encryption ensures another layer of security for the system. Every information on the Blockchain is hashed cryptographically. So, changing or trying to tamper with the data means changing all the hash IDs. If someone wants to corrupt the network, he/she would have to alter every data stored on every node in the network. There could be millions and millions of people, where everyone has the same copy of the ledger.

Distributed Ledgers :

A public ledger will provide every information about a transaction and the participant nodes. Many people can see what really goes on in the ledger. The ledger on the network is maintained by all other users on the system. Distributed ledger responds really well to any suspicious activity or tamper. Nodes act as verifiers of the ledger. If a user wants to add a new block others would have to verify the transaction and then give the green signal.To make the blockchain features work, every active node has to maintain the ledger and participate for validation.

Consensus :

Every blockchain succeeds because of the consensus algorithms. Every blockchain has a consensus to help the network make any transactions. In simple, the consensus is a decision-making process for the group of nodes active on the network. Here, the nodes can come to an agreement quickly and relatively faster. When millions of nodes are validating a transaction, a consensus is absolutely necessary for a system to run smoothly. Nodes might not trust each other, but they can trust the algorithms that run at the core of it.

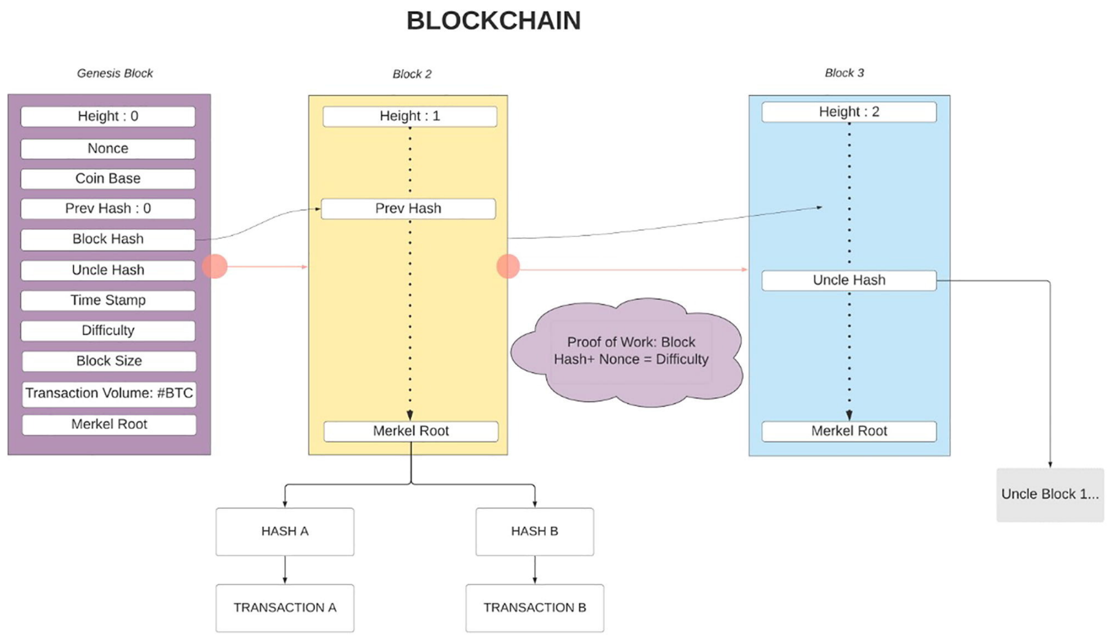

Figure 5.2: Structure of Block in BlockChain

5.2.2 Types of BlockChain:

There are four types of Blockchain

Public BlockChain

: A public blockchain is a non-restrictive, permission-less distributed ledger system. A node or user which is a part of the public blockchain is authorized to access current and past records, verify transactions or do proof-of-work for an incoming block and do mining.

Private BlockChain

A private blockchain is a restrictive or permission blockchain operative only in a closed network. Private blockchains are usually used within an organization or enterprises where only selected members are participants of a blockchain network. The level of security, authorizations, permissions, accessibility is in the hands of the controlling organization. Thus, private blockchains are similar in use as a public blockchain but have a small and restrictive network.

Consortium BlockChain

A consortium blockchain is a semi-decentralized type where more than one organization manages a blockchain network. More than one organization can act as a node in this type of blockchain and exchange information or do mining.

Hybrid BlockChain

A hybrid blockchain is a combination of the private and public blockchain. It uses the features of both types of blockchains that is one can have a private permission-based system as well as a public permission-less system. With such a hybrid network, users can control who gets access to which data stored in the blockchain. Only a selected section of data or records from the blockchain can be allowed to go public keeping the rest as confidential in the private network. The hybrid system of blockchain is flexible so that users can easily join a private blockchain with multiple public blockchains. A transaction in a private network of a hybrid blockchain is usually verified within that network.

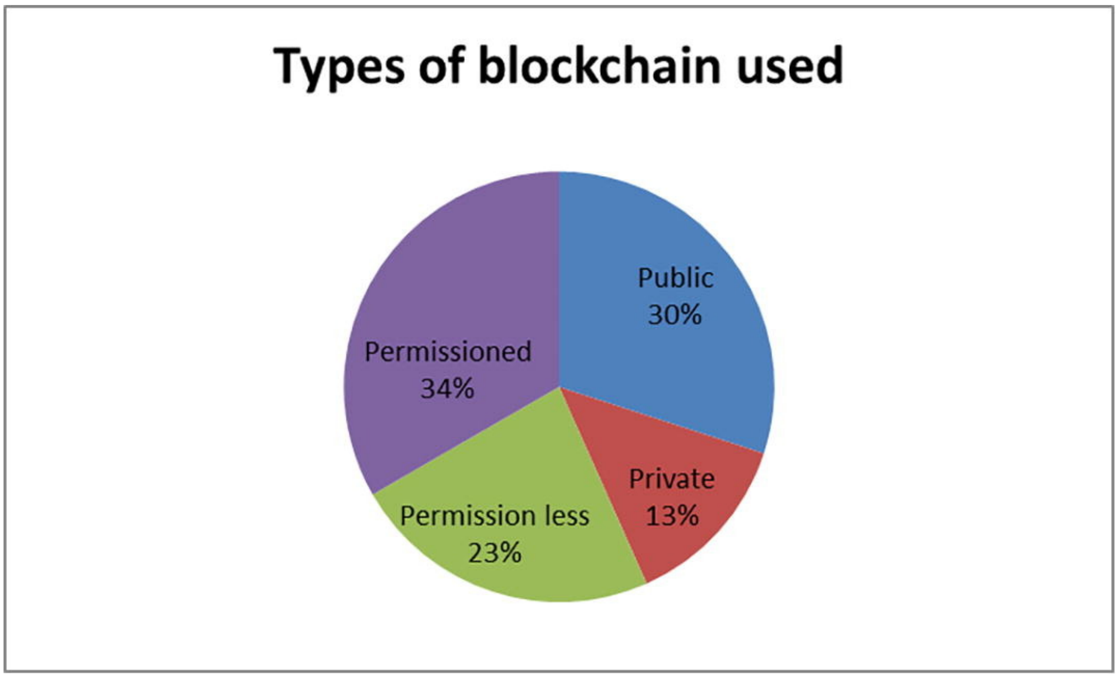

Figure 5.3: Types of BlockChain

5.3 Smart Contract

A smart contract is a self-executing contract with embedded rules that defines the agreement and conditions between two or more parties.Currently, there are more than 30 million smart contracts deployed worldwide spread across industries such as BFSI, energy, life sciences, public sector, and real estate.

Developing a smart contract requires thorough and systematic analysis to understand the best and most efficient parameters needed for coding an effective contract.Parameters are important as they shape the codes and rules for auto-execution.

So basically, Smart contracts are self-executing programs that are stored on Blockchain and run when predefined conditions are met. They are the digital versions of the standard paper contract, whose

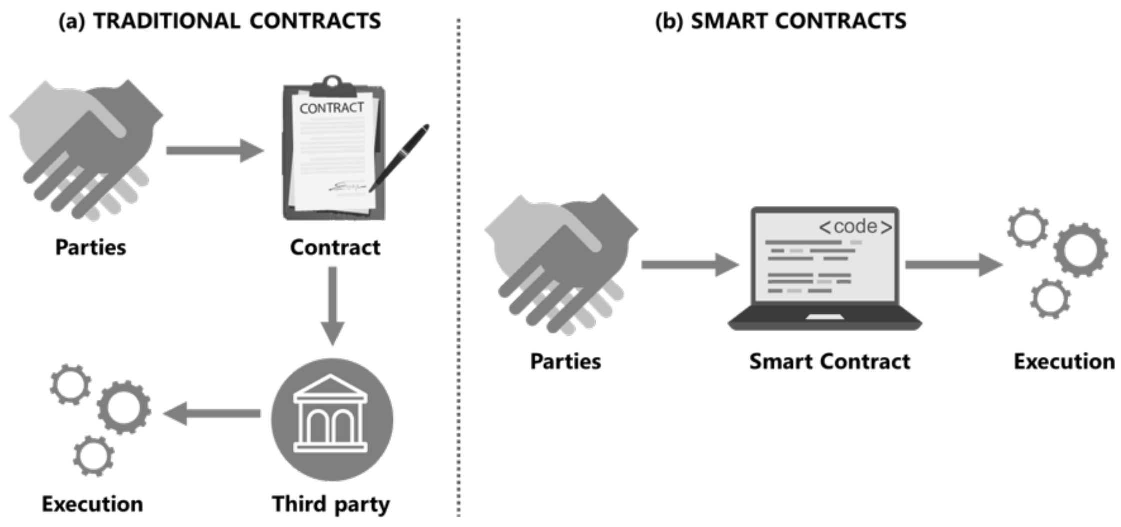

Figure 5.4: Traditinol and Smart Contracts

terms of the agreement are written in the lines of code. They automatically execute transactions if certain conditions are met without requiring the help of a third party to manage or approve the transaction. This third party could be a government organization, a lawyer, or any other entity. For instance, in traditional paper contracts, a document outlines the terms of conditions between two parties, which is enforceable by law. If one party A violates the terms, party B can take Party A to court for not complying with the agreement. Whereas, in the smart contract, such agreements are written in code, so the conditions of the agreement are automatically enforced without any third party getting involved.

5.3.1 How do smart contracts work?

The life cycle of smart contracts consists of four consecutive phases: creation, deployment, execution, and completion.

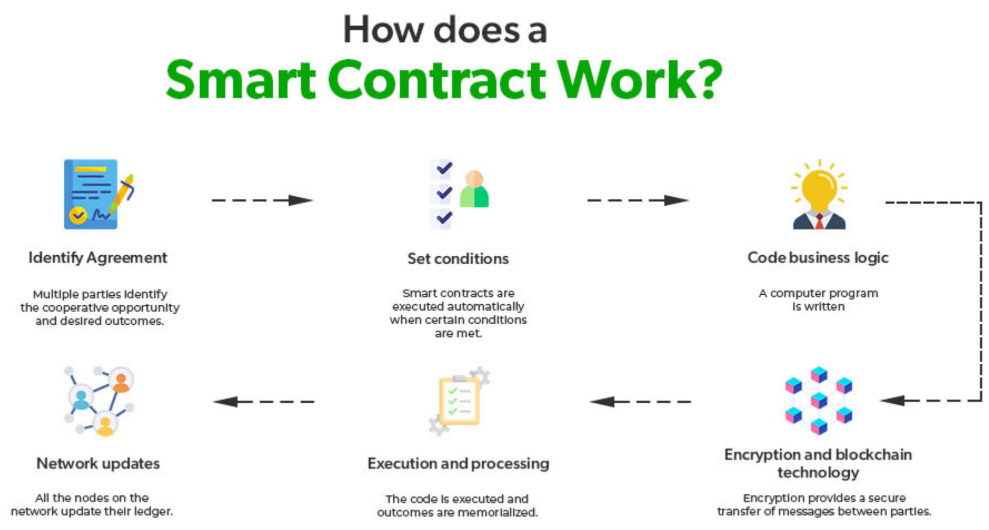
How does a
Smart Contract Work?
Identify Agreement
Multiple parties identify the cooperative opportunity and desired outcomes.
Set conditions
Smart contracts are executed automatically when certain conditions are met.
Code business logic
A computer program is written
Encryption and blockchain technology
Encryption provides a secure transfer of messages between parties.
Execution and processing
The code is executed and outcomes are memorialized.
Network updates
All the nodes on the network update their ledger.

Figure 5.5: Abstract About smart contract working through diagram

Creation of smart contracts:

Before any transaction, the contractual parties determine the terms of the contract. It is important to note that lawyers or counselors may be required to help parties draft an initial contractual agreement, but for the execution of the agreement, a third party is not required. After the contractual terms and conditions are finalized, they are translated into programming code called a smart contract. Basically, the code represents a number of different conditional statements that describe the possible scenarios of a future transaction. They work by following simple "if. . . then. . . else" statements that are written into code on a Blockchain. The "if" statement is the most basic conditional statement that is used to make a decision on whether the statement or code will be executed or not. "If" the condition is true, "then" the statement will be executed, while if the condition is false, then else statement will be executed. For instance, in Fig, if A is true, then B will be executed, else C will be executed.

Deployment of smart contracts:

When the smart contract is created, the parties can agree to the terms and conditions by applying their digital signature on the contract. A digital signature is a cryptographic technique that binds a person to digital data. Private keys of the parties are used to create digital signatures. Digital signature has been explained in detail in Part 6. Then the smart contract is stored on the Blockchain network, which can not be altered/modified, making them immutable. Any correction or revision in the contract requires the creation of a new contract. For that reason, you must pay special attention to writing and testing code to avoid introducing bugs in the contract that will never be fixed.

Because of storing the smart contracts on Blockchain, they become decentralized. It means that smart contracts are not controlled by a single machine/human. In fact, all the nodes on the Blockchain store the same contract with exactly the same state. Additionally, the digital signature + the public key of the parties are enough for nodes to verify that the private keys associated with the parties have been used to make signatures on the contract.

Moreover, during this phase, any transfers to the smart contract's receiving wallet address are blocked. For instance, a smart contract is signed between buyer and supplier. According to the contract, fund transfer from the buyer's wallet to the supplier only occurs once the buyer receives the goods from the supplier. As a result, any sort of funds transfer on the supplier's wallets will be blocked. The nodes act as a governing body that validates whether the predefined conditions for contract execution have been satisfied.

Execution of smart contracts:

After the deployment of smart contracts, the contractual terms are monitored and evaluated by all the Blockchain nodes in the network. Once the predefined conditions are met, the smart contract is self-executed. The funds in the form of coins are released from the buyer's wallet to transfer them to the supplier (as a commitment of exchanging goods). The released funds create a transaction triggered by the met criteria. Consequently, the executed transaction is validated by the nodes on the Blockchain to ensure fulfillment of the contract conditions. This verification process is done by the Proof-of-Work or Proof-of-Stake consensus mechanisms. The verified transactions and the updated state of smart contracts are then stored on the Blockchain.

Completion of smart contracts:

According to the predefined conditions on the smart contract, after the buyer receives the goods from the seller, the seller's wallet is unlocked. Therefore, the funds get transferred from the buyer to the supplier's wallet. This marks the completion of the smart contract, which is then closed and recorded on the Blockchain.It is important to note that a sequence of transactions has been executed during the deployment, execution, and completion phases of a smart contract. Therefore, all three phases need to write data to the blockchain.

5.3.2 Advantages of smart contracts

Trust:

One of the most significant benefits that smart contracts have over traditional contracts is that they are automatically executed when the agreement conditions are met. There is no need to wait for a third party to execute them. In other words, smart contracts remove the need for trust. For example, if an employer wants to write a

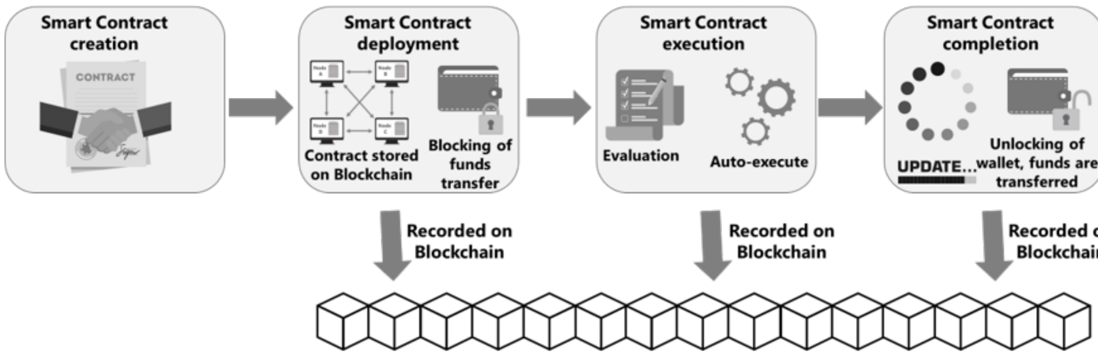

Figure 5.6: Working Architecture of Smart-Contract

contract that holds funds in escrow for a hired person, allowing them to withdraw funds only after the completion of an assigned task. In this case, a third party like Upwork initiates the contract and holds the payment in escrow. Once the hired person completes the work, the employer reviews the work. If the employer is satisfied, he notifies Upwork, and the payment is released. For this, both parties have to rely on the reputation of the intermediaries like Upwork. On the other hand, the smart contract would work in a fully automated way, ensuring that the hired person receives the money after completing the task. And the employer gets the task completely done. For this, the parties don't have to rely on intermediaries. Once conditions have been met, nodes on the Blockchain validate the transaction, and the money is released to the hired person. Smart contracts can also hold funds. For example, you could write a smart contract that holds funds in escrow for a child, allowing them to withdraw funds after a specific date. If they try to withdraw the funds before the specified date, the smart contract won't execute.

Transparency:

Another property of smart contracts that makes them so critical is the transparency they bring along with them. As discussed earlier, smart contracts contain a detailed list of terms and conditions agreed upon by the parties involved. This pre-agreed setting eliminates the chances of issues and disputes at later stages as the terms and conditions are proposed and passed by the parties themselves. These terms and conditions remain visible to every party involved in the transaction. Thus bringing transparency in the system. Issues related to communication gaps are also reduced with the implementation of smart contracts as there is only a single version of the truth that is visible to everyone on the network.

Immutability:

Smart contracts are immutable, which means their code and conditions can not be changed or updated once they are deployed on the Blockchain. Moreover, they are stored and duplicated throughout the whole distributed Blockchain system, and thus are traceable and auditable. As a result, malicious behaviors like financial fraud can be greatly mitigated. If you want to change an existing smart contract, you will have to deploy a new version of the contract.

Better Time Efficiency:

Another significant benefit of implementing smart contracts is better efficiency. In the traditional system, with the paperwork involved, it generally takes days to get a request processed. There is a lot of unnecessary duplication of documents at various stages of the process. Further, the involvement of a large number of intermediaries makes the process more complex, cumbersome, and time-consuming. However with the implementation of smart contracts, all these redundant and unnecessary steps are eliminated from the process, which significantly reduces the time taken to complete the transactions.

Safety and security:

Smart contracts along with Blockchain are tamper-proof, reliable, and secure. This characteristic of security and safety brings in more value and trust in the associated transactions.

5.3.3 Solidity:

On Ethereum, smart contracts are written in a programming language called Solidity.

5.4 Zero-knowledge proof

A Zero-Knowledge Proof blockchain is a sophisticated combination of cryptographic privacy techniques and blockchain technology, providing a platform where transactions and interactions can be verified and recorded without compromising the confidentiality of the data involved.

ZKPs are cryptographic methods that allow one party (the prover) to prove to another party (the verifier) that a certain statement is true without revealing any information beyond the validity of the statement itself.

Zero-knowledge proofs aren't exclusive to blockchain. They can be used in many areas where privacy is important. For example, online identity verification processes, secure voting systems, and confidential data sharing in businesses can all use ZKPs to protect personal information while still ensuring that necessary verification takes place.

Blockchain is a distributed ledger technology known for its transparency, immutability, and security. It's a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system.In blockchain transactions, ZKPs allow the validation of transactions without exposing sensitive details. This integration transforms a blockchain into a more private and secure platform, enhancing its utility in a wide range of applications where confidentiality is as crucial as integrity and transparency.

Example Let's say Denise wants to prove to her bank that she has enough income for a loan but doesn't want to reveal her exact salary.

Denise's employer issued her an employment credential as a verifiable credential that she manages in her digital wallet on her phone. The credential includes her legal name, salary, job title, employee number, the date she started at the company, and her work email. Because the credential integrates Zero-Knowledge Proof technology, she can prove her salary to the bank without revealing the details.

5.4.1 What Is a Zero-Knowledge Proof Blockchain?

A Zero-Knowledge Proof blockchain is a sophisticated combination of cryptographic privacy techniques and blockchain technology, providing a platform where transactions and interactions can be verified and recorded without compromising the confidentiality of the data involved.ZKPs are cryptographic methods that allow one party (the prover) to prove to another party (the verifier) that a certain statement is true without revealing any information beyond the validity of the statement itself. Zero-knowledge proofs aren't exclusive to blockchain. They can be used in many areas where privacy is important. For example, online identity verification processes, secure voting systems, and confidential data sharing in businesses can all use ZKPs to protect personal information while still ensuring that necessary verification takes place.
Blockchain technology in zkp: Blockchain is a distributed ledger technology known for its transparency, immutability, and security. It's a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system.In blockchain transactions, ZKPs allow the validation of transactions without exposing sensitive details. This integration transforms a blockchain into a more private and secure platform, enhancing its utility in a wide range of applications where confidentiality is as crucial as integrity and transparency.
Zero Knowledge Proof Blockchain Example: This example shows how privacy is maintained with a Zero-Knowledge Proof blockchain. Let's say Denise wants to prove to her bank that she has enough in-

come for a loan but doesn't want to reveal her exact salary. Denise's employer issued her an employment credential as a verifiable credential that she manages in her digital wallet on her phone. The credential includes her legal name, salary, job title, employee number, the date she started at the company, and her work email. Because the credential integrates Zero-Knowledge Proof technology, she can prove her salary to the bank without revealing the details. When the bank sends her an email with a QR code, which is requesting the salary verification, Denise uses her digital wallet to present the credential that proves that she satisfies the income requirement without showing her exact salary.

ZKP: As a concept, ZKPs are about proving knowledge without revealing any of the information. The key aspect here is that the verifier learns nothing beyond the fact that the prover knows the secret, so no details of the secret are disclosed. This is ideal for situations where revealing any detail is unnecessary or undesirable.

For example, in a confidential voting system, such as for corporate board decisions or private organizations, voters can prove that their vote was cast correctly without revealing who or what they voted for.

Zero-Knowledge Proof Technology Enabling Selective Disclosure: ZKP technology with selective disclosure is a more sophisticated application of ZKP principles in use cases where some level of disclosure is required or beneficial. The prover can share specific information as needed without exposing irrelevant information on a credential.

For example, a job applicant may need to prove they hold certain qualifications or degrees without revealing their full educational history or grades. Selective disclosure enables them to share just the necessary credential information, like the degree title and issuing institution, while withholding other details.

5.4.2 Zero Knowledge Proof Algorithm:

An algorithm is like a recipe in a cookbook, but instead of making food, it's designed to solve problems or perform computations. It's a series of instructions that are followed to complete a task. These instructions are clear, unambiguous, and designed to be executed by a computer.

In the context of ZKPs, the algorithm is a special kind of procedure used for proving that one party (the prover) has certain information to another party (the verifier), without actually revealing the information itself.This ensures privacy and security in various applications, including blockchain and authentication.

5.4.3 The Benefits of Combining Zero-Knowledge Proofs With Blockchain

This is an example of selective disclosure where someone can choose to only present relevant information on their credential to a verifier.

1. **Enhanced Privacy:** Typically, blockchain technology is praised for its ability to maintain a public ledger that is transparent and immutable. However, this transparency can sometimes be a double-edged sword where the details of the transactions are open to everyone. But with a Zero-Knowledge Proof blockchain, ZKPs are used to validate transactions without revealing their contents.

2. **Security:** ZKPs add an additional layer of security to the blockchain by ensuring that even if the blockchain is publicly accessible, sensitive information remains confidential. The impact of ZKP on blockchain privacy and security is significant. It not only ensures that data remain confidential but also minimizes the risk of sensitive information being exploited, a concern that is increasingly common in our digital age.

3. **Selective Disclosure:** Zero-Knowledge proof technology enables selective disclosure of information. Users can prove cer-

tain aspects of their transactions or identity (such as age or residency) without revealing any other irrelevant personal information on their credentials.

4. **Decentralized Identifiers (DIDs):** DIDs are globally unique identifiers that enable a person or entity to prove ownership of their identity without relying on a centralized authority. A DID is a random string of letters and numbers. Each DID is associated with a DID document that contains public keys and other information necessary for authentication. When DIDs are registered on a blockchain, the associated identity information and public keys are stored in a tamper-resistant manner. This immutability ensures that the DID remains trustworthy and cannot be easily manipulated. When a DID is used to issue Verifiable Credentials, the fact that it is registered on a blockchain enhances trust. Verifiers can independently check the blockchain to ensure that the DID is legitimate and associated with a trusted entity, which strengthens the trust in the credentials issued by that DID. Unlike some blockchain systems, the Dock blockchain never ever stores any personally identifiable information or verifiable credentials. This approach significantly enhances privacy and security by minimizing exposure to potential data breaches.

5.4.4 Zero-Knowledge Proof technology enables these powerful privacy functions that can serve a variety of use cases:

- **Selective Disclosure:** Selective Disclosure is a privacy tool that allows users to share selective information within a credential instead of presenting all of the details on the credential. For example, a resident can show that she is eligible for a local government service by presenting information that proves that she lives in a qualified city without showing irrelevant details like her email address or date of birth.

- **Range Proofs:** A range proof is a method that allows someone to prove a number or value within a specific range without revealing the actual number or value. For example, a government program is providing grants to people between the ages of 25 to 35 to learn how to code and get into the tech industry. Applicants can prove that they fall within this age range without disclosing their date of birth.

- **Verifiable Encryption:** Verifiable encryption is a way to securely share your information and only trusted parties can access the actual details when needed. It's like using a special lock to secure your information that only certain people such as regulators have the key to open when it's necessary. For example, confidential legal documents such as wills or non-disclosure agreements are stored in a digital vault. Only the involved legal representatives or, under certain conditions, a court of law can view the details.
- **Threshold Anonymous Credentials:** This is a method where credentials are co-issued by a group of entities. Instead of one person issuing a credential, a group of parties have to work together to provide it. For the credential to be issued, a certain number of parties have to agree.
- **Custom Conditions:** Credential verifiers can check to see if specific credential details meet the required conditions (either a number of other types of values) without seeing the actual details. A user could set up requirements where an exact match must be met to satisfy the condition or a cumulative sum adds up to a minimum or maximum amount among such as 12 months of pay slips is enough for a lower income worker to apply for a small loan.

5.5 Zk-Snark and ZoKrates

The Blockchain is designed for openness and transparency. Pseudonymity is protected by delinking addresses to real world identities, but blockchain transactions are designed for active mapping, monitoring and follow up where one wishes. In a system without trust, this transparency serves as a self examination of confidence. For example, BItcoin transactions are anonymous — but only as far as ownership of wallets are unknown — the actual transactions are viewable to the public in order to be auditable.

In order to increase adoption, blockchains need to have greater privacy guarantees. Current security measures are inadequate: for example, preventing addresses from being linked to identities such as coin tumblers; Indeed, some users of the blockchain are often looking for a link to identity, but also offer a degree of privacy. Businesses want discreet negotiation with manufacturers and individuals who have a large net worth are trying to keep their assets secure.

ZkSNARKs allow the possibility to verify correctness of a computation without having to execute it on its own. Importantly, it is not necessary to know the content of the calculation itself in order to be able to verify that calculations have been made correctly. This allows the effective implementation of zkSNARKs, when privacy concerns arise; information is not to be disclosed in order that something can be verified without creating a distrustful environment.

5.5.1 What is a zk-SNARK?

zk-SNARK stands for "Zero-Knowledge Succinct Non-Interactive ARgument of Knowledge" and is a method of constructing a proof, where one can prove possession of information or completion of a computation/transaction without said information transferred or said computation computed again to be verified. Verification involves a "prover" showing a generated zk-SNARK proof to the "verifier", who verifies said proof for correctness and completion — without knowing the information from the original zk-SNARK proof process. Therefore, there is no significant trust needed and only one party needs to have the information in order for the transaction to proceed.

- **Zk (Zero Knowledge):** Touched upon earlier, this means that the verifier knows nothing about the computation or data itself when verifying the proof, apart from ensuring that the statement/transaction/computation is valid or true. Particularly important is that the verifier knows nothing about the witness (see below for "knowledge").
- **S (Succinct):** This is of great importance, particularly when dealing with complex transactions. The size of the proofs is very small, even for complex computations. This has large implications for increasing blockchain network capacity and efficiency.
- **N (Non Interactive):** This is also very important for the efficiency of the protocol. Interactivity refers to rounds of messages being sent between provers and verifiers. In other systems of verification numerous rounds of exchange between a verifier and prover are necessary to ensure correctness. In zk-SNARKs there is only an initial set up phase (a single point of contact). SNARKs have a property called "public verifier" so anyone can verify without interacting multiple times, which is both resource and time-intensive. This makes zk-SNARKS conducive to use on blockchains.
- **AR (Arguments):** Statements are theoretically computational sound. That is, proofs are reflective of true statements. This is based on the fact that creating logically incorrect proofs is computationally infeasible, while correct proofs may be generated relatively efficiently.
- **K (Knowledge):** It is not possible for the prover to construct a proof without knowing a "witness," or a set of inputs that the computation being verified is run on. Examples include, the

address that the values are spent from, the path to a certain node in a Merkle tree, a hash function preimage, etc. As mentioned earlier, for the zero-knowledge aspect to hold true, although the prover knows the witness, it is important that the verifier has no knowledge of the witness.

5.5.2 How is a zk-SNARK Implemented?

Now that we know what the acronym stands for, there are few elements of implementing zk-SNARKs:

- **Succinctness, through Random Sampling:**
- **Encoding, as a Polynomial Problem:**
- **Encryption:**
- **Zero-Knowledge:**
- **Succinctness, though Random Sampling:**

Encoding will explain succinctness a bit more, but we noted earlier that this succinctness is an important aspect of SNARKs that allow small, and thus easily verifiable proofs for transactions.The steps to achieve encoding are as such:

Computation or Code → Arithmetic Circuit → R1CS (Rank 1 Constraint System) → QAP (Quadratic Arithmetic Program) → zk-SNARK

- The first step of the process is the circuit, which breaks down all steps to small operations of addition, subtraction, multiplication, or division. The rank 1 constraint system, or R1CS, confirms that the values travel in the right direction (inputs travel "left to right"), where the R1CS will confirm the value a+b is correct.

- In R1CS (rank 1 constraint system), the verifier checks almost every constraint. This is resource and time intensive, so as mentioned above in succinctness, it is possible to bundle all constraints into a representation called the QAP "Quadratic Arithmetic Program" where the single constraint is now checked. This constraint is between polynomials, where one checks that two polynomials match at a single randomly chosen point, which will correctly verify the proof to a high degree of certainty.
- Encryption: An encoding or encryption function "E" is used that has some homomorphic properties. Homomorphic encryption allows computations on ciphertexts (the result of encrypted plaintext using an algorithm called a cipher), which generates an encrypted or scrambled result that, when decrypted, matches the result if they had been performed on plaintext. This allows the prover to compute the values E(t(s)), E(h(s)), E(w(s)), and E(v(s)) without knowing the value of s. Instead, only E(s) and other encrypted values are known. This once again demonstrates the ability to shield private information and values.
- Zero-Knowledge: In zero-knowledge proofs, the prover has knowledge of something called a witness, which satisfies certain parameters and is kept hidden. The prover's duty is to convince the verifier that he or she has this witness without revealing the parameter. Through this prover-verifier model through a witness, no data is transferred or revealed in the verification of the computation or ownership of information, only that it is correct. This is combined with the zero-knowledge aspect, because the values above E(t(s), E(h(s), E(w(s), and E(v(s)) can be hidden even further by multiplying with another number. Thus, a verifier can still check that the structure that has been encrypted is correct, without knowing the encoded values which are masked.

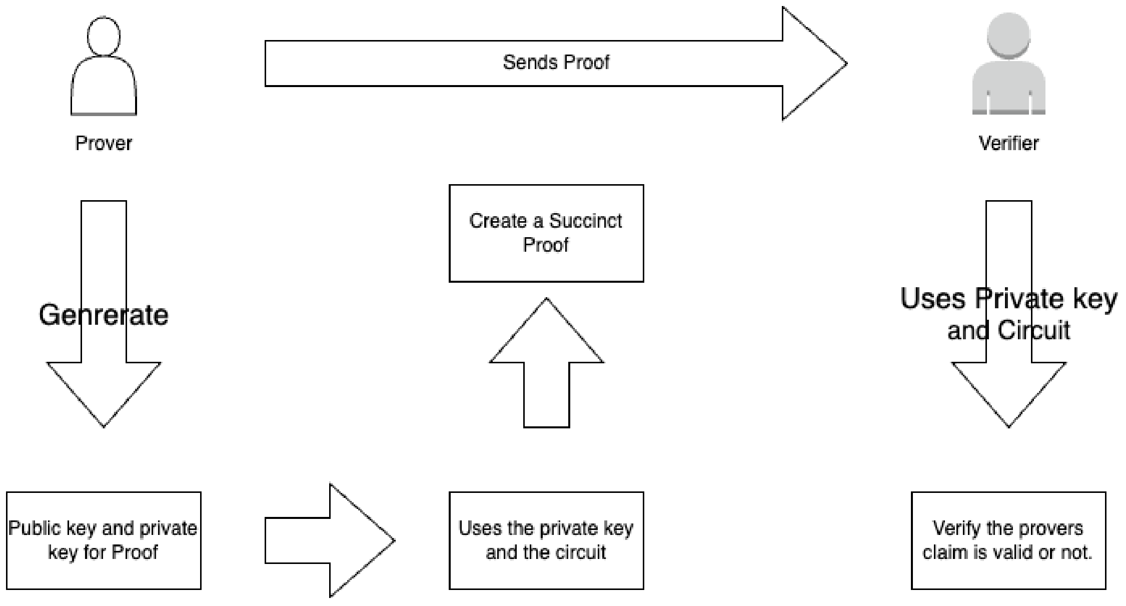
Prover
Sends Proof
Verifier
Genrerate
Create a Succinct Proof
Uses Private key and Circuit
Public key and private key for Proof
Uses the private key and the circuit
Verify the provers claim is valid or not.

Figure 5.7: How Zk-Snark Works:

5.5.3 Applying Zk-SNARKs on Ethereum: The ZoKrates Toolbox:

Zk-SNARKs' potential on Ethereum achieve three major goals and lend themselves to interesting applications in smart contracts.First, it allows increased privacy, where verifiers can prove on information that does not need to be revealed — where in the current system all is needed to be public. Second, it allows efficiency, with only a short, one-way interaction needed (refer to succinct, whereas currently multiple rounds of communication are needed) where the complexity of verification is independent of the complexity of the computation that is being proved, which allows standardization of costs. Lastly, it allows scalability, a major issue facing most blockchains. Zk-SNARKs have wide potential, but at the moment it is very difficult to implement them on the Ethereum network and in smart contracts — this is where ZoKrates comes in.With scalability an issue on Ethereum, off-chain solutions have become popular. There are currently two ways of making sure off-chain computations such are correct.

One method is publishing the result of the computation, and verification is to work backwards from that, which is a solution some off-chain protocols use. The second, which zk-SNARKs use, is that when doing the computation, to generate a proof that proves that the computation was correct. Then, all that is needed is to take a version of the result on-chain and validate the proof on-chain. As mentioned earlier, this is the prover-verifier relationship, where the proof is generated during the zk-SNARK implementation off chain, and the verifier validates it on-chain at a single point.Off-chain, a proving and verifying key are created and provers use a prover's key to create an off-chain proof. The smart contract can then be executed, and the verification done on-chain using the proof, verifying key, and other parameters. If the outcome is correct, the smart contract is executed and other on-chain activity can follow.

An example of zk-SNARKs in smart contracts is showing an ID at a bar. Suppose there is a smart contract that is executed allowing entrance if an ID scanned demonstrates the owner is over 21. If there is a hash of the ID containing the birthdate on the blockchain, a prover can prove that he or she has an ID with a birthdate that is over 21 by hashing the ID off-chain and provide the zk-SNARK proof that will the ID will hash to that of the value on-chain, all without revealing the actual ID. The verification is cheap and done right in the EVM (Ethereum Virtual Machine) with one iteration.

So how does Zokrates work? ZoKrates is a high level, non-turing complete language (as opposed to Ethereum, which is turing complete) and a compiler which compiles a set of conditions as such:
First form R1CS (rank 1 constraint system; a list of conditions) $\rightarrow$ QAP quadratic arithmetic program -tree with only addition/multiplication $\rightarrow$ zk work to generate a prover and verifier.
This is explained above in the elements of a zk-SNARK. One can

learn more by reading the Zokrates paper. The ZoKrates library creates the actual ethereum smart contract for the user — the only part specified is what constraints are wanted, written in the ZoKrates language. One can then create proving and verifying keys for a circuit and, so long as the user has a valid "solution" or "witness" to the circuit, one can generate a valid transaction to the contract. This setup closely ties to the elements of zk-SNARK implementation explained earlier.

5.6 Some Genreal Question-Answers

1. **Q. What initiatives have been taken for IdM using blockchain?**
   Ans. The government organizations are using permissioned blockchain as a platform to store digital identities. Estonia government has been using blockchain since 2014 to preserve national data. The 'ID2020 alliance' has established a multistakeholder partnership to ensure digital ID. Blockchain has also been applied in education for credit management of students.

2. **Q. What challenges still need to be addressed when using blockchain for identity management?**
   Ans. The challenges that still need to be addressed when using blockchain for identity management include achieving anonymity at all levels, giving users control over their private information and access, protecting user identities and personal information, and providing technological development to control and monitor identities.

3. **Q. Why is there a need for an analytical evaluation and assimilation of existing literature in the field of IdM using blockchain technology?**
   Ans. There is a need for an analytical evaluation and assimilation of existing literature in the field of IdM using blockchain technology because the studies published so far have not covered

important aspects such as identity management initiatives using blockchain technology, the popularity of consensus mechanisms, papers published between 2009 and 2016, and ongoing research projects. The researchers' curiosity about blockchain has created a demand for a comprehensive understanding of the existing literature. Systematic mappings, although time-consuming, provide clear and thorough information on the research in progress. This evaluation will help identify key frameworks, address gaps, and suggest opportunities for future researchers.

4. **Q. What are the major issues that need to be addressed in the future regarding secure identity management using blockchain?**
   Ans. The major issues that need to be addressed in the future regarding secure identity management using blockchain include scalability, interoperability, cost associated with each transaction, data protection and authentication, data storage and replication, risk management, and the lack of skilled employees.

5. **Q. What are the frameworks of IdM using blockchain?**
   Ans. The frameworks of IdM using blockchain include Bitnation, which is an open source governance platform based on Ethereum smart contract.

6 Existing Claim Model For DIMS

We shall first explain some of the terms associated with an identity model. And then we describe and mathematically define the existing claim identity model and the improved claim identity model respectively.

6.1 Terminologies

1. **Entity:** The entities are physical or logical objects with distinct existence, which include individuals and collectives (e.g., companies, governments, banks). According to their functionalities, entities are divided into three categories, namely identity providers (IdPs), service providers (SPs), and users.
2. **Attribute:** An attribute is a characteristic of an entity, which consists of a name-value pair. And a qualified attribute for authorization must have two properties: Possession and Endorsement.
3. **Identifier:** An identifier is an attribute whose value can be used to uniquely identify an entity within a context (Ferdous et al., 2014). In the identity solutions based on blockchain, the blockchain address usually serves as the identifier, which is controlled by the blockchain private key.
4. **Claim:** The claim, usually composed of IdP's digital signature and stored in the smart contract, is a verifiable endorsement for the attribute.

5. **Hashclaim:** The hashclaim is a hash value of the user's public key, the attribute identifier, and a salt, which can be interpreted as the verifiable attestation that the possession of attribute has been transferred to the owner of the public key included in hash-claim's preimage.
6. **Identity:**The identity of an entity is defined as the set of attributes bound to its identifier and corresponding claims or hash-claims. Specially, we assume that the attribute identifier is bound to itself by default but it collects an empty set of claims and hashclaims.
7. **Profile:**When the user wants to access the service from the SP, he/she needs to share the required attributes. For the sake of privacy, the asserted identifier and attributes for authorization are defined as the profile.

6.2 Already existing claim identity model

Here, The user first self-generates many attributes that do not compromise privacy, as seen in Fig., and then publishes them in the blockchain using the blockchain private key that governs his or her identity. The publisher's address and the attribute, which could be interpreted as the attribute's possession, will be stored in the smart contract.

Subsequently, the IdPs issue claims based on the characteristics of eligible users using the blockchain private key. The issuer's address and the claim, which is a verifiable endorsement, will be kept in the smart contract.

The user merely needs to prove that they are the owner of the identification in order to use SP's services; typically, this may be done by transferring funds from their blockchain account wallet. Then SP may retrieve the entire identification.

A: the set of attributes, D: the set of IdPs, U: the set of users and

Identifier_u: the identifier of the user u. a claim, which is issued on attribute a by the IdP d, can be represented by $(C_a)^d$

For an attribute $\alpha \in A$, the set of claims issued about α, denoted as AC_α, is defined as follows:

$$AC_\alpha = \{d_\alpha^1, d_\alpha^2, \ldots, d_\alpha^m\} \quad (d_1, d_2, \ldots, d_m \in D) \tag{6.1}$$

Since not every attribute can receive claims from IdPs, AC_α could be an empty set.

For a user $u \in U$, the set of attributes possessed by u is defined as A_u. And the identity of u, denoted as $identity_u$, is defined as follows:

$$identity_u = \langle \text{identifier}_u, \{(a, AC_a) \mid a \in A_u\}\rangle \tag{6.2}$$

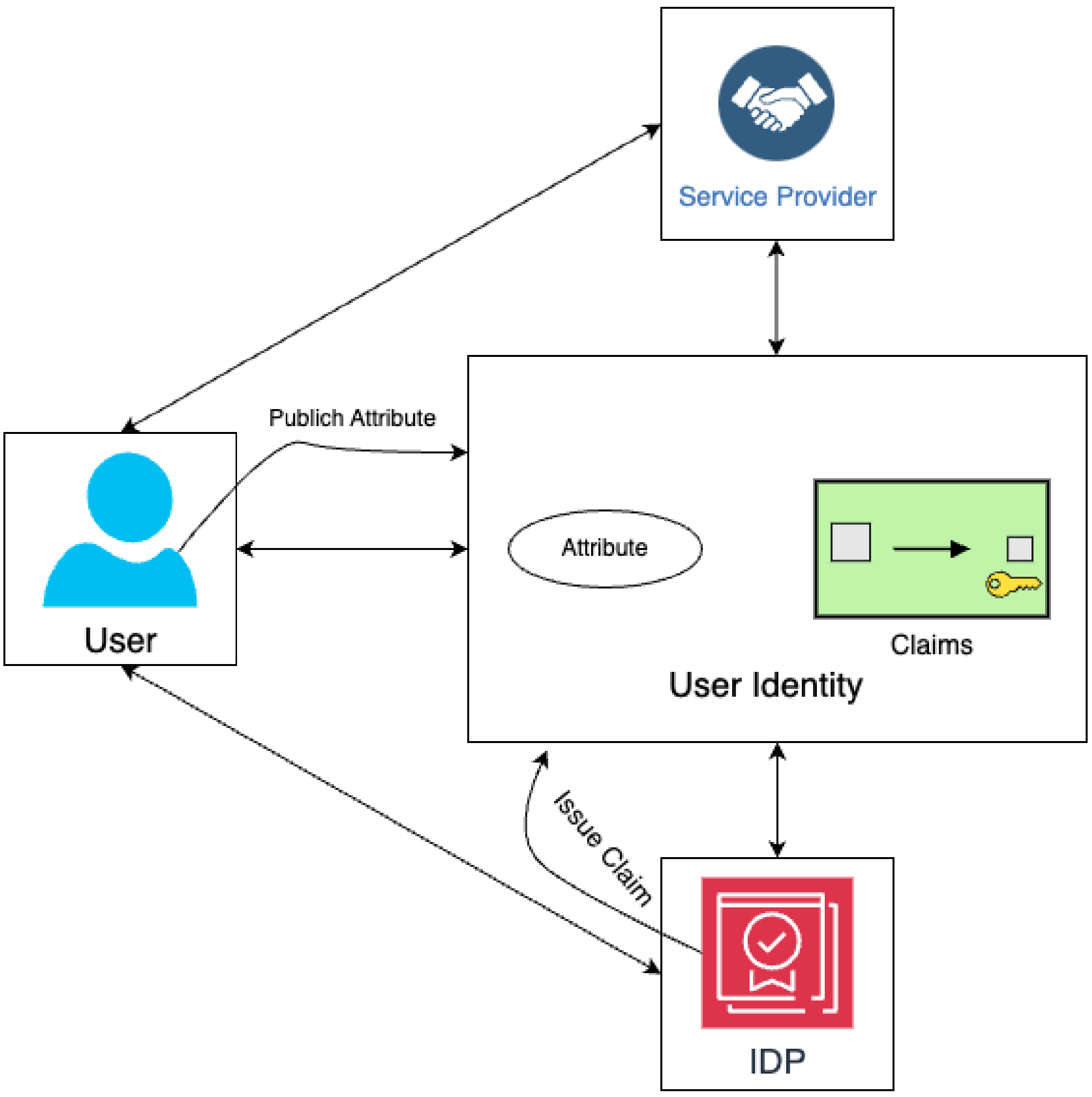

Figure 6.1: Existing Claim Identity Model

6.3 Improved Claim identity model

As shown in fig, in the improved claim model, the privacy attribute containing a name-value pair is created by IdP, then the smart contract will store the mapping of the attribute to IdP's identifier. Hence, the IdP's operation of publishing the privacy attribute could be regarded as the endorsement of the privacy attribute.

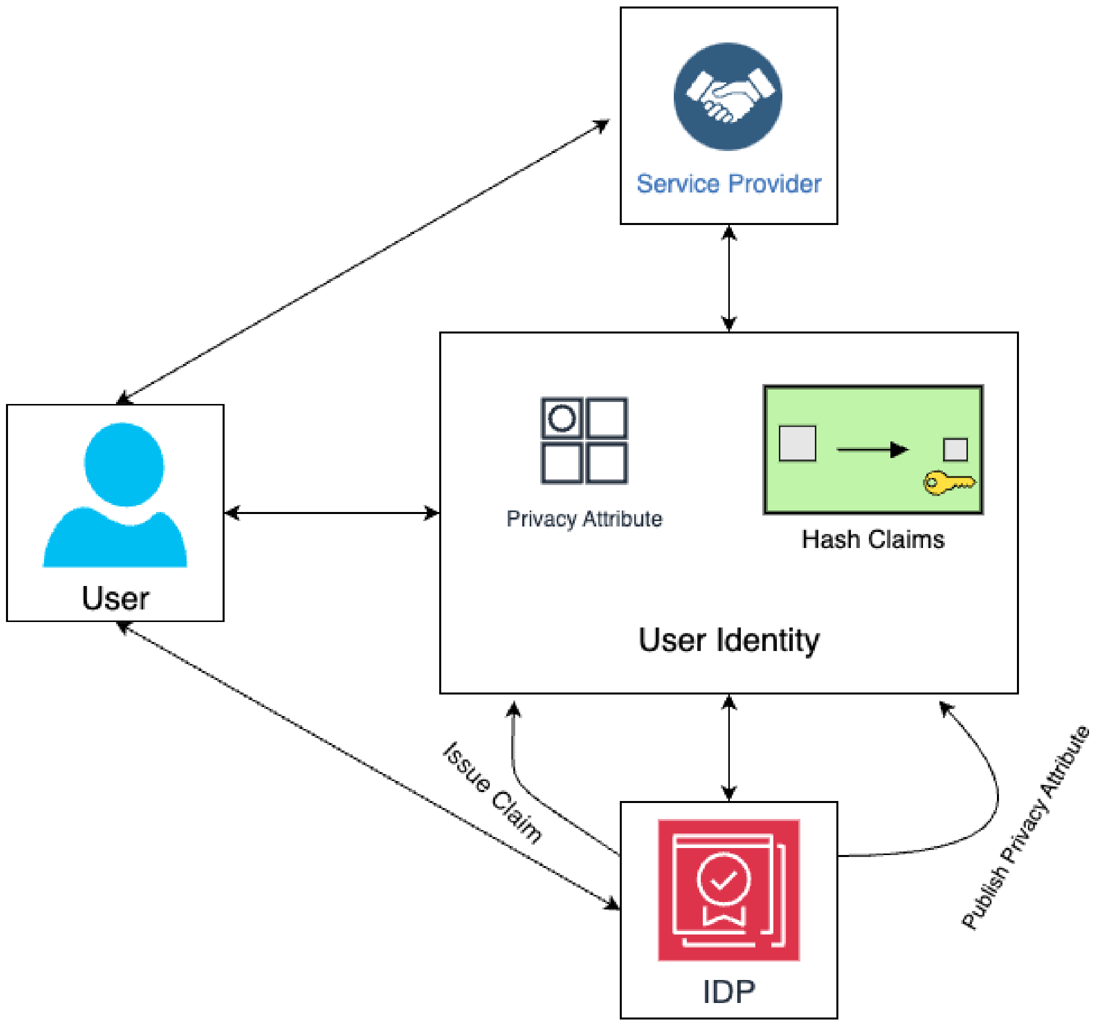

Figure 6.2: Improved Claim Identity Model

the IdP's operation of publishing the privacy attribute could be regarded as the endorsement of the privacy attribute. And The hashclaim attests that possession of this attribute. After that, the IdP issues a hashclaim about this privacy attribute by ZKP. The hashclaim attests that possession of this attribute has been transferred to the user, which will be stored in the smart contract. The process of issuing the hashclaim will not disclose the user's public key. From the perspective of other entities in blockchain, IdP just issues a hashclaim on a self-created attribute, but other entities did not know which user the IdP transferred the possession of this attribute to. When the user wants to access the service from SP, he/she could assert the ownership of the privacy attribute. Since the endorsement of privacy attribute could be checked in smart contracts, the user next needs to attest to the possession of attribute and identifier via ZKP. To distinguish privacy attributes from non-privacy attributes of the prior model, we assume that $\mathcal{M}$ denotes the set of privacy attributes. Furthermore, a hashclaim, which is issued on the privacy attribute m by IdP d, can be represented by h_m, whose preimage contains the user's public key.

For a user $u \in U$, the set of privacy attributes possessed by u is defined as M_u. Thus the identity of u is defined as follows:

$$identity_u =< identifier_u, (m, h_m) \mid m \in M_u, d \in D > \tag{6.3}$$

6.4 Comparision between both Claim Identity Model

The relation between user, identifier and identity is shown in Fig. And the solid arrows indicate that the relation is globally visible, and the dotted arrows indicate that the relation is not visible to other entities.

It could be found that each entity uses one blockchain address that is controlled by a private key to uniquely identifies itself in both two blockchain-based identity models. However, in the existing claim identity model, the user's action to publish attributes in blockchain has exposed his/her possession of these attributes. The endorse-

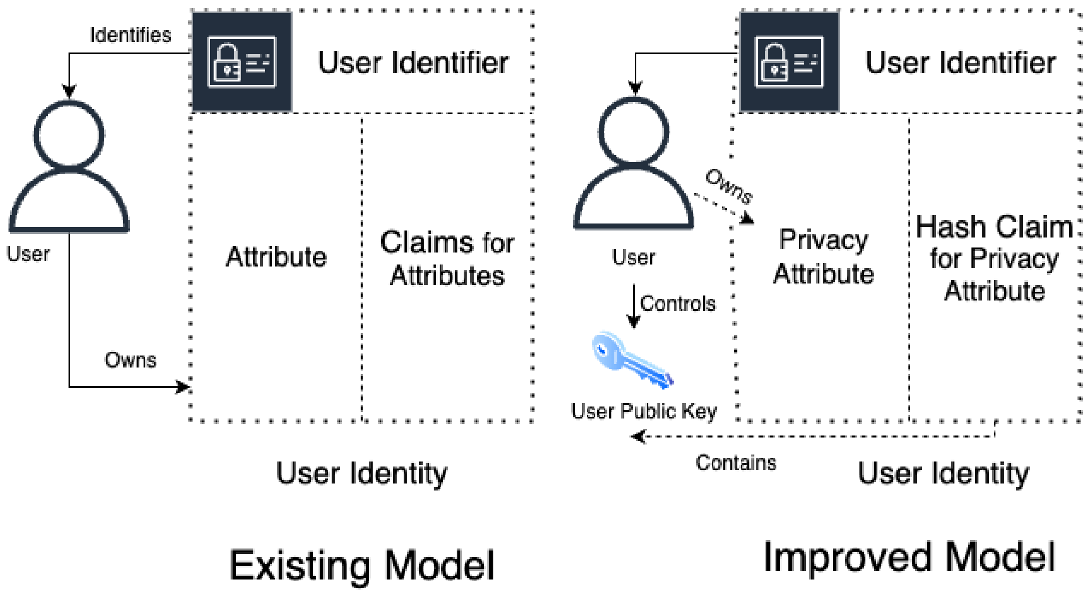

Figure 6.3: Comparision Between both Models

ment for attribute comes from the claim issued by IdPs. Therefore, the ownership between user and identity in existing claim identity model is visible to other entities as shown in Fig. By contrast, in the improved claim identity model, the endorsement for the privacy attribute comes from IdPs' action to create it. By leveraging ZKP, user's possession of the attribute is hidden in the verifiable hash-claim, whose preimage includes the user public key that is invisible. Thus what attributes the user has is also unknown as shown in Fig, realizing effective privacy protection. In addition, when accessing the service from SPs, the user in the existing claim model only needs to assert and attest to the fact that he/she possesses a certain identifier. But the user in the improved model needs to follow the challenge-response protocol to assert that he/she possesses a certain identifier and a set of required attributes and demonstrate that preimages of the corresponding hashclaims include his/her public key by ZK-SNARK.

7 Proposed Architecture and Secheme Overview

To implement the improved claim identity model, we propose a Blockchain-and-ZKP-based digital identity management system with the help of zk-SNARK and Ethereum. And the proposed system model is presented in Fig.

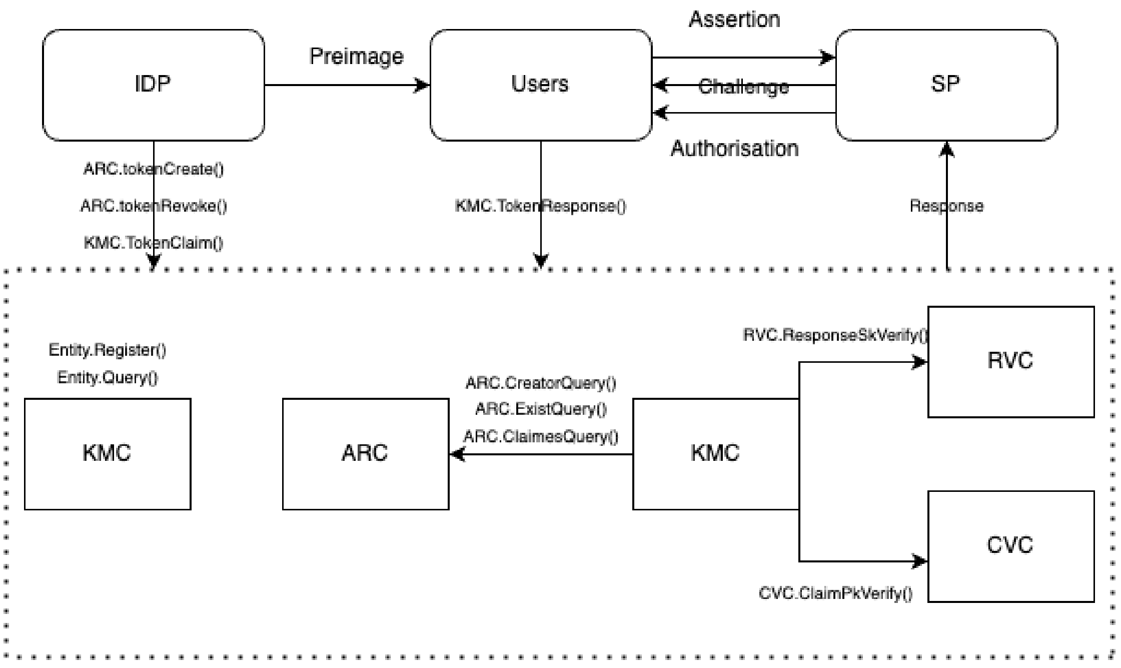

Figure 7.1: Architecture Of Proposed Schema

There are three entities in our scheme, namely Identity provider, Service provider and User.

1. **Identity Provider (IdP):** Responsible for publishing and revoking the privacy attributes, issuing verifiable hashclaims on them.
2. **Service Provider (SP):** Responsible for providing online or offline services whose access policy contains requirements for attributes and validating the profile provided by the user.
3. **User:** Possessing identity attributes stored in blockchain, accessing the service from IdPs by proving ownership of his/her identifier and attributes.

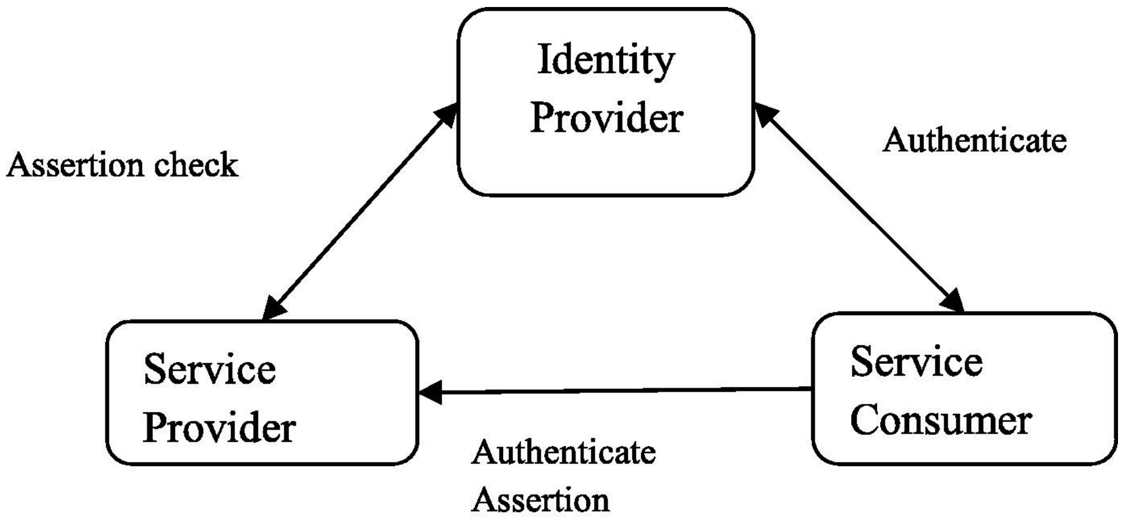

Figure 7.2: Cycle of 3 Entities

The privacy attributes are abstracted as tokens in here. Besides, five smart contracts are developed to achieve the full life cycle of the privacy attributes including:

1. **Entity Register Contract (ERC)**
2. **Attribute Repository Contract (ARC)**
3. **Knowledge Management Contract (KMC)**
4. **ClaimPK Verify Contract (CVC)**

5. **ResponseSK Verify Contract (RVC)**

ERC provides an interface entityRegister() for entities to register public keys and an interface entityQuery() for querying entity public key information. In addition, four functions designed to be called by entities to manage attributes are:

- ARC.tokenCreate()
- ARC.tokenRevoke()
- KMC.tokenClaim()
- KMC.tokenResponse()

The function call relationship between contracts is shown in Fig upside, where KMC will call the functions in ARC, RVC, and CVC to manage attributes. In addition, since ZoKrates cannot directly derive the public key from the Ethereum private key based on secp256k1, we use sha256 in the standard library of ZoKrates instead of secp256k1. We define the key pair generated by sha256 as a ZK key pair, where the ZK public key will be included in the hashclaim's preimage, and the ZK private key needs to be properly kept.

Before the solution runs, our DIMS needs to deploy smart contracts and execute the zk-SNARK setup phase to share the proving key with all. The entities need to register their ZK public key withblockchain address in ERC. As shown in Fig, the challenge-response protocol consists of five steps, and the full life cycle of privacy attributes includes the following four procedures:

- **Creation:** Creation is the initial step in which a privacy attribute token that contains a name-value pair is published by the IdP and the ERC will record the creator's address as the endorsement of the attribute.
- **Transferring:** To secretly transfer the possession of the privacy attribute to the user in blockchain, IdP needs to issue the hashclaim on this token and prove that it performed the particular

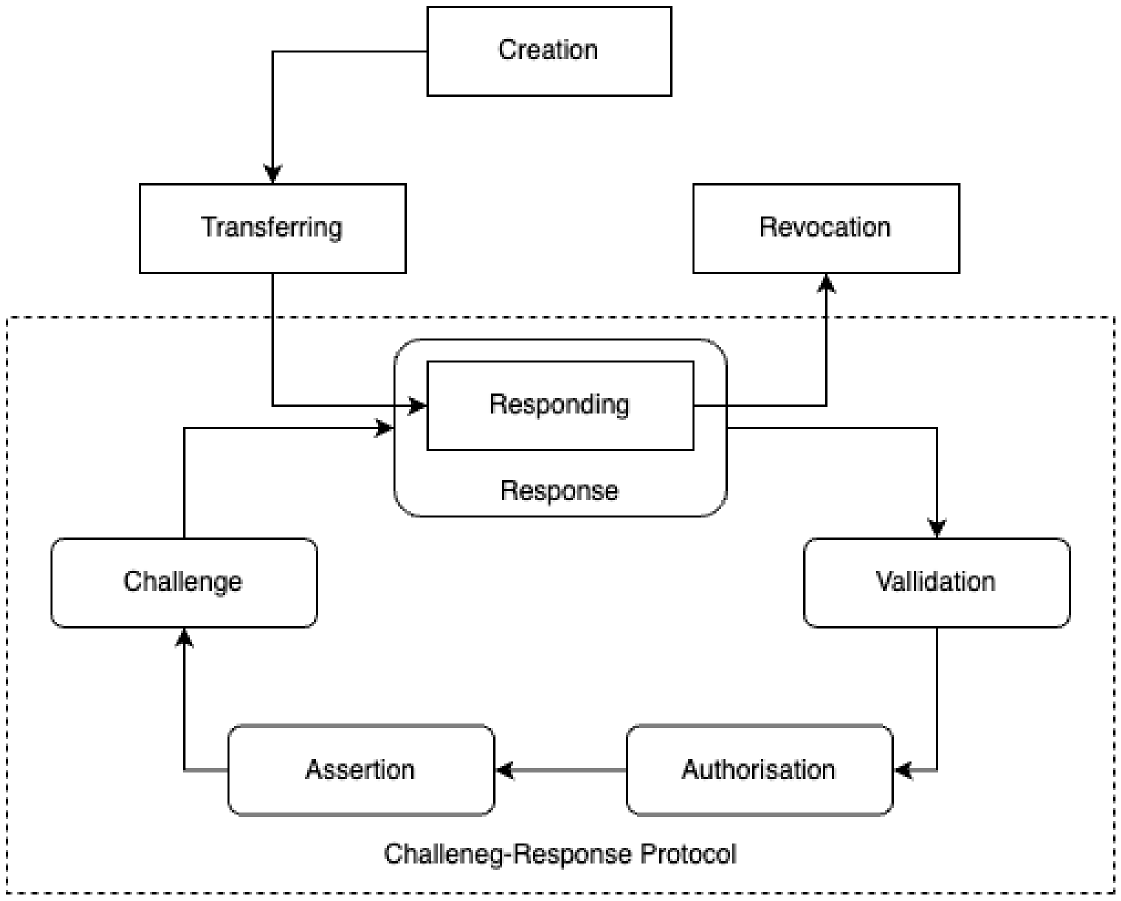

Figure 7.3: the challenge-response protocol and the full life cycle of privacy attributes

computation named claimPK that hashclaim is hashed from the token identifier and user ZK public key by ZKP.

- **Responding:** To demonstrate the possession of the privacy attribute token, the user needs to prove that he/she owns the ZK private key corresponding to the ZK public key contained in the pre-image of hashclaim. The particular computation performed by the user during this process is named responseSK. This step is also a part of the challenge-response protocol.

- **Revocation:** The final step is the revocation process which allows the IdP to block the responding of the attribute by modifying the existence field of the token.

Used notations and functions are listed respectively in Tables 1 and 2.

Full life cycle of privacy attributes:

7.1 Phase 1: Creation:

To create the privacy attribute token, the IdP needs to follow the creation procedure as presented in Fig.

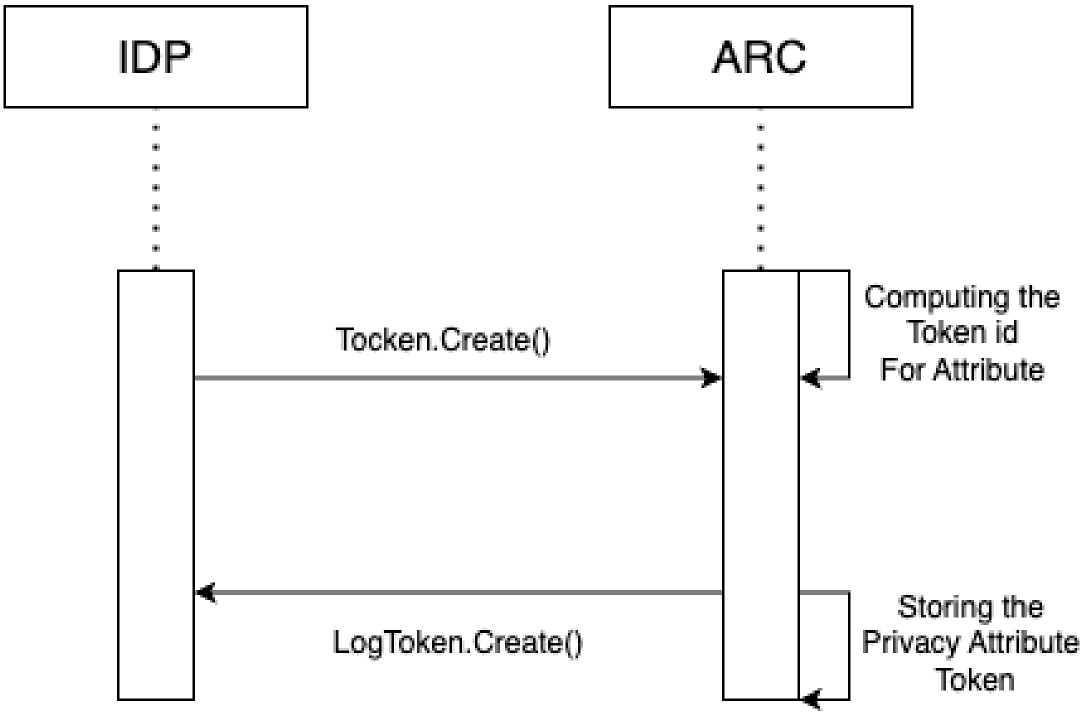
IDP
ARC
Tocken.Create()
Computing the Token id For Attribute
LogToken.Create()
Storing the Privacy Attribute Token

Figure 7.4: Sequance Diagram for Creation

First, the IdP sends a transaction to call tokenCreate() in ARC according to the metadata format shown in Table 3. The transaction is signed by the blockchain private key sk^E_{idp} that controls the blockchain address E_{idp}, so that the sender of the transaction could be recorded as the publisher of the privacy attribute token. As described in Algorithm 1, the tokenCreate() will compute a unique 'tokenId' for each privacy attribute token for retrieval and store the mapping from tokenId to attribute.

Algorithm 1 tokenCreate()

```
  function TOKENCREATE(_name, _value, _hash)
      Inputs: _name, _value, _hash
      Outputs: LogTokenCreate()
      tokenId ← sha256(msg.sender —— timestamp —— _name —— _value —— _hash)
      attrTokens[tokenId].name, attrTokens[tokenId].value, attrTokens[tokenId].hash ←
  _name, _value, _hash
      attrTokens[tokenId].creator ← msg.sender
      attrTokens[tokenId].exist ← True
      attrTokens[tokenId].claimed ← False
      Emit event LogTokenCreate(tokenId, _name, _value, msg.sender, _hash)
  end function
```

Given the cost of storing data in blockchain is very expensive, and storing large data will affect the system throughput and block synchronization speed, the scheme allows off-chain storage (such as IPFS) for large data (such as the PGP key) to optimize the system performance while providing the 'hash' to ensure the attribute authenticity. The field of 'name' is used to record the attribute type, such as role, certificate, qualification, or marked as IPFS. The field of 'value' is responsible for recording the content of the attribute or the IPFS address. The field of 'creator' is responsible for recording the publisher of the attribute, which could be regarded as a verifiable endorsement from the IdP. The field of 'exist' is True once the token is created. When the token is revoked, it will be set to False to block the token's response. The field of 'claimed' is False when the token is created. Only after the IdP completes the subsequent process of issuing the hashclaim on this token, the field of 'claimed' will be set to True. And this token cannot be issued the hashclaim again if 'claimed' is True, preventing the IdP from transferring the attribute possession to multiple users. Last, the creation of the privacy attribute token will trigger the event LogTokenCreate() to notify the IdP completion status. At this time, the IdP has completed the creation, but the attribute possession has not been transferred to the qualified user.

7.2 Phase 2: Transferring

7.2.1 Particular computation of claimPK:

To transfer the possession of attribute created in the phase of creation to the user u, the IdP needs to include the identifier of the privacy attribute token and the ZK public key of u into the preimage of the hashclaim. To prevent the attacker from obtaining the ownership between u and the attribute by enumerating ZK public keys registered in ERC and token identifiers to crack the hashclaim, an additional token salt figure is also contained in the preimage.

We assume that the identifier of the privacy attribute token is τ, the ZK public key queried from ERC is $(pk^Z)_u$, the token salt figure is ε_τ, and the hashclaim as the sha256 result of these three values is H_τ. The `claimPK` could be abstracted to the following formula:

$$H_\tau = \text{sha256}(\tau || (pk^Z)_u || \varepsilon_\tau) \tag{7.1}$$

The $(pk^Z)_u$ and ε_τ are private inputs, τ and H_τ are public inputs of `claimPK`. Zokrates will perform the following calculations:

```
1. Function verifyClaimPK(pubInp_claim, priInp_claim)
   • $H'_\tau \leftarrow \text{sha256}(\tau || (pk^Z)_u || \varepsilon_\tau)$
   • if $H'_\tau == H_\tau$ then
       – return True
   • else
       – return False
```

7.2.2 The procedure of transferring:

The set of constraints of claimPK Cst_{Claim} has already been compiled locally. The generators for zk-SNARK witnesses and proofs are abstracted to witnessGen() and proofGen() respectively within Zokrates. The procedure of issuing the hashclaim can be summarized as presented in Fig.

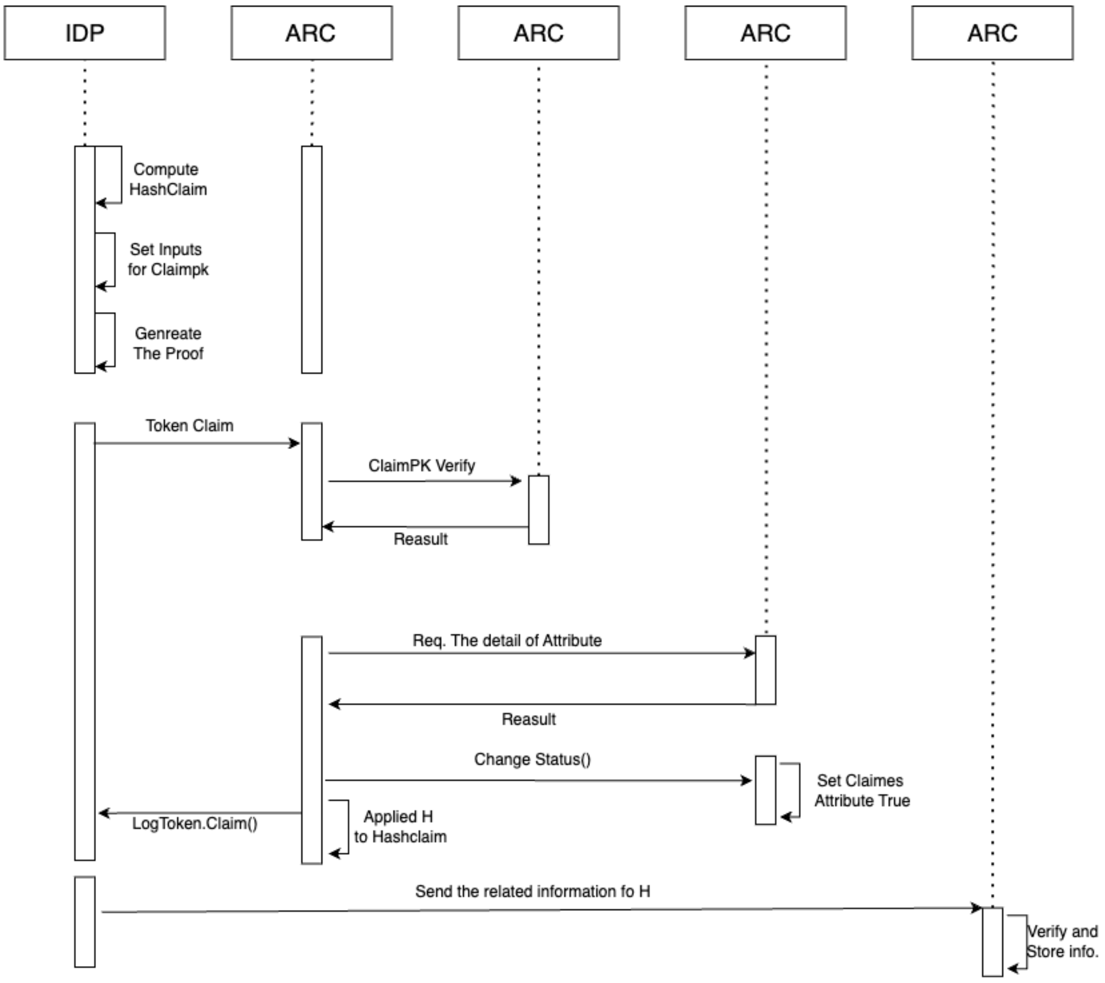

Figure 7.5: Sequence diagram of responding.

First, the IdP computes hashclaim $H'_\tau := \text{sha256}(\tau||(pk^Z)_u||\varepsilon_\tau)$ and feeds public inputs $pubInp\_claim = (\tau, H_\tau)$ and private inputs $pubInp\_claim = ((pk^Z)_u, \varepsilon_\tau)$ for the circuit of claimPK to generate witness $\psi_{\text{claim}} = \text{witnessGen}(\mathcal{C}_{\text{claim}}, (pubInp\_claim, priInp\_claim))$.

According to claimPK proving key p_claim, the IdP generates the zk-SNARK proof: $\sigma_{\text{claim}} = \text{proofGen}(p\_claim, \psi_{\text{claim}})$.

The IdP then calls the `tokenClaim()` in KMC by sending a transaction signed by $(sk^E)_{\text{idp}}$ from address E_{idp} to verify the $pubInp\_claim$ and σ_{claim}.

The details of ‘tokenClaim()‘ are shown in Algorithm 2.

We assume that the identifier of the privacy attribute token is τ, the ZK public key queried from ERC is $(pk^Z)_u$, the token salt figure is ε_τ, and the hashclaim as the sha256 result of these three values is H_τ. The `claimPK` could be abstracted to the following formula:

$$H_\tau = \text{sha256}(\tau || (pk^Z)_u || \varepsilon_\tau) \tag{7.2}$$

The $(pk^Z)_u$ and ε_τ are private inputs, τ and H_τ are public inputs of `claimPK`. Zokrates will perform the following calculations:

```
function VERIFYCLAIMPK(pubInp_claim, priInp_claim)
    $H'_\tau \leftarrow \text{sha256}(\tau || (pk^Z)_u || \varepsilon_\tau)$
    if $H'_\tau == H_\tau$ then
    return True
    else
    return False

end function
```

The procedure of transferring:

The set of constraints of claimPK Cst_{Claim} has already been compiled locally. The generators for zk-SNARK witnesses and proofs are abstracted to witnessGen() and proofGen() respectively within Zokrates. The procedure of issuing the hashclaim can be summarized as presented in Fig.

First, the IdP computes hashclaim $H'_\tau := \text{sha256}(\tau || (pk^Z)_u || \varepsilon_\tau)$ and feeds public inputs $pubInp\_claim = (\tau, H_\tau)$ and private inputs

$priInp\_claim = ((pk^Z)_u, \varepsilon_\tau)$ for the circuit of claimPK to generate witness $\psi_{\text{claim}} = \text{witnessGen}(\mathcal{C}_{\text{claim}}, pubInp\_claim, priInp\_claim)$.According to claimPK proving key p_{claim}, the IdP generates the zk-SNARK proof $\sigma_{\text{claim}} = \text{proofGen}(p_{\text{claim}}, \psi_{\text{claim}})$.The IdP then calls the `tokenClaim()` in KMC by sending a transaction signed by $(sk^E)_{\text{idp}}$ from address E_{idp} to verify the $pubInp\_claim$ and σ_{claim}.

The details of `tokenClaim()` are shown in Algorithm 2.

Algorithm 2 tokenClaim()

```
function TOKENCLAIM(pubInp_claim, $\sigma_{claim}$)
    Inputs: pubInp_claim, $\sigma_{claim}$
    Outputs: LogTokenClaim()
    verificationResult ← CVC.claimPKVerify( $\sigma_{claim}$, pubInp_claim)
    if verificationResult ≠ True then
        Revert
    else
        Require ARC.creatorQuery( $\tau$ ) == msg.sender
        Require ARC.claimedQuery( $\tau$ ) == False
        Require ARC.existQuery( $\tau$ ) == True
        Append $H_\tau$ to the ever-increasing list claimList
        Set the field of ‘claimed’ of $\tau$ to True
        Emit the event LogTokenClaim(msg.sender, $\tau$, $H_\tau$)
    end if
end function
```

Then, the IdP watches the triggered event `LogTokenClaim()`. If the event is successfully emitted, the IdP will store the related information $\{\tau, E_u, (pk^Z)_u, \varepsilon_\tau\}$ in the local database. Finally, the IdP sends the preimage of H_τ to u through the secure channel. The user u follows the following steps to confirm preimage information and stores $\{\tau, H_\tau, \varepsilon_\tau\}$ in the local database:

- Compute sha256($\tau || (pk^Z)_u || \varepsilon_\tau) \rightarrow H_\tau$.
- Check H_τ is already in the list `claimList`.
- Check the ’claimed’ of τ is True.
- Check the ’exist’ of τ is True.

This process successfully transfers the attribute possession to u, while ensuring that the possession relation is hidden by zk-SNARK, instead of being public in smart contracts.

7.3 Phase 3: Responding.

7.3.1 Particular computation of responseSK:

To attest to the possession of the privacy attribute token, the user u needs to prove that he/she owns the ZK private key corresponding to the ZK public key contained in the preimage of hashclaim. In addition, to ensure that the proof can only be used once, we design a one-off responding waste, whose preimage includes ZK private key and different response salt figures to distinguish from previous wastes, effectively avoiding replay attacks.

The preimage of H_τ held by the user is $\{\tau, (pk^Z)_u, \varepsilon_\tau\}$. We assume that the user's ZK private key is $(sk^Z)_u$, the 128-bit response salt figure is δ, and the responding waste as the sha256 result of these two values is N_δ. The `responseSK` can be abstracted to the following formulas:

$$(pk^Z)_u = \text{sha256}((sk^Z)_u)$$
$$H_\tau = \text{sha256}(\tau || (pk^Z)_u || \varepsilon_\tau)$$
$$N_\delta = \text{sha256}((sk^Z)_u || \delta)$$

The $(sk^Z)_u$, δ, and ε_τ are private inputs, τ, H_τ, and N_δ are public inputs. Zokrates will perform the following calculations:

1. Compute $(pk^Z)'_u = \text{sha256}((sk^Z)_u)$
   The newly calculated value $(pk^Z)'_u$ should be equal to $(pk^Z)_u$, but Zokrates does not need to set $(pk^Z)_u$ as a private input. The correctness of $(sk^Z)_u$ will be verified by $(pk^Z)'_u$. The correctness of $(pk^Z)_u$ will be verified in step c.

2. Compute $H'_\tau = \text{sha256}(\tau || (pk^Z)_u || \varepsilon_\tau)$

3. Check $H'_\tau == H_\tau$.
   The correctness of τ is verified by the contract. Note that H'_τ calculated in (b) should be equal to H_τ, and the correctness of H_τ will be verified by the contract. Namely, when feeding the correct ZK private key and token salt figure, H'_τ should be equal to H_τ and exist in the claimList, ensuring the correctness of private inputs $\{(sk^Z)_u, \varepsilon_\tau\}$.

4. Check $\text{sha256}((sk^Z)_u||\delta) == N_\delta$
   Step d guarantees that when feeding the correct private input, N_δ should be equal to $\text{sha256}((sk^Z)_u||\delta)$, ensuring the correctness of N_δ.

5. Return True.

In this way, the private input is checked against at least one public input in each check phase. By constructing the circuit in this way, the prover can only share the public input and proof, which will be verified against the verification key stored in the contract.

7.3.2 The procedure of responding:

The set of constraints of `responseSK` $\mathcal{C}_{\text{response}}$ has already been compiled locally. The evidence that can only be decrypted and verified by the SP is called the responding content. The detailed procedure of responding is described in Figure.

First, the user u computes the responding waste:

$$N_\delta = \text{sha256}((sk^Z)_u||\delta)$$

and feeds public inputs:

$$\text{pubInp\_response} = (\tau, H_\tau, N_\delta)$$

and private inputs:

$$\text{priInp\_response} = ((sk^Z)_u, \varepsilon_\tau, \delta)$$

for the circuit of `responseSK` to generate witness:

$$\psi_{\text{response}} = \text{witnessGen}(\mathcal{C}_{\text{response}}, \text{pubInp\_response}, \text{priInp\_response})$$

Then, u generates the zk-SNARK proof:

$$\sigma_{\text{response}} = \text{proofGen}(p_{\text{response}}, \psi_{\text{response}})$$

Against the `responseSK` proving key p_{response} in Zokrates, u calls the `tokenResponse()` in KMC by sending a transaction from anonymous addresses u to verify pubInp_response and σ_{response} and enable ES_{π} to be triggered in the event `LogTokenResponse()`.

Algorithm 3 KMC.tokenResponse()

```
function TOKENRESPONSE(pubInp_response, $\sigma_{response}$, ES_$\pi$)
    Inputs: pubInp_response, $\sigma_{response}$, ES_$\pi$
    Outputs: LogTokenResponse( $\tau$, ES_$\pi$)
    verificationResult ← RVC.responseSKVerifiy( $\sigma_{response}$, pubInp_response)
    if verificationResult == False then
        Revert
    else
        Require ARC.existQuery( $\tau$ ) == True
        Require ARC.claimedQuery( $\tau$ ) == True
        Check $H_{\tau}$ is already in the list claimList
        Check $N_{\delta}$ is not in the list revocationList
        Append $N_{\delta}$ to the ever-increasing list revocationList
        Emit the event LogTokenResponse( $\tau$, ES_$\pi$)
    end if
end function
```

ES_{π} is the attestation of possession of the identity identifier, which is elaborated later in the report. The details of `tokenResponse()` are shown in Algorithm 3. The waste will be appended to `revocationList` to prevent replay attacks. Since the length of the response salt figure is 128 bits, each privacy attribute can perform responding for 2^{128} times. It can be considered that users do not have to worry about exhausting the responding times.

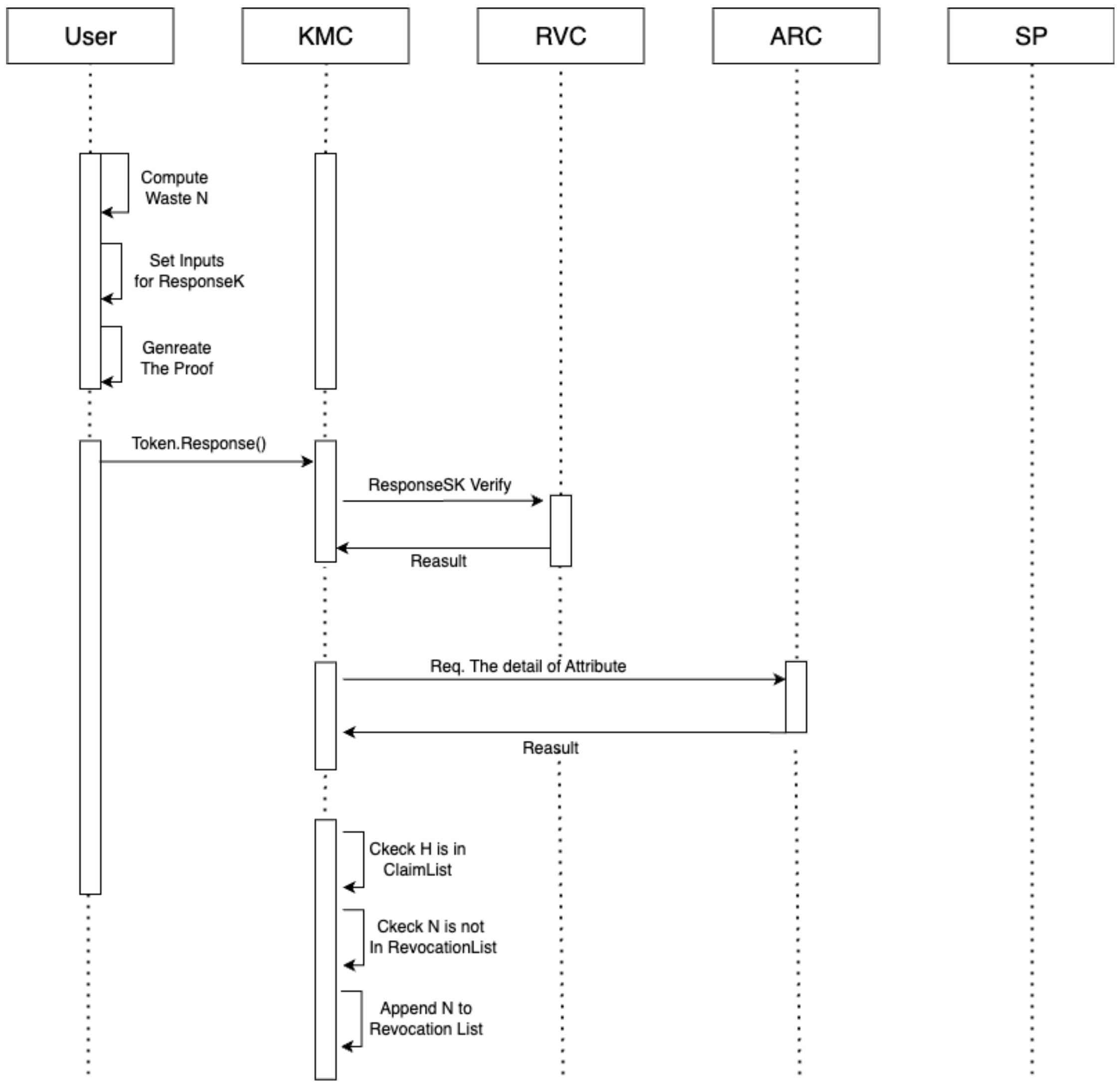

Figure 7.6: 1D Discrete Random walk[9]

7.4 Phase 4: Revocation.

As shown in Figure , to revoke the privacy attribute token τ, the IdP calls `tokenRevoke()` in ARC from blockchain address E_{idp} recorded in 'creator'. The details of `tokenRevoke()` are described in Algorithm 4. This function will set the 'exist' of τ to False to block functionality of `tokenClaim()` and `tokenResponse()` to complete the revocation of τ.

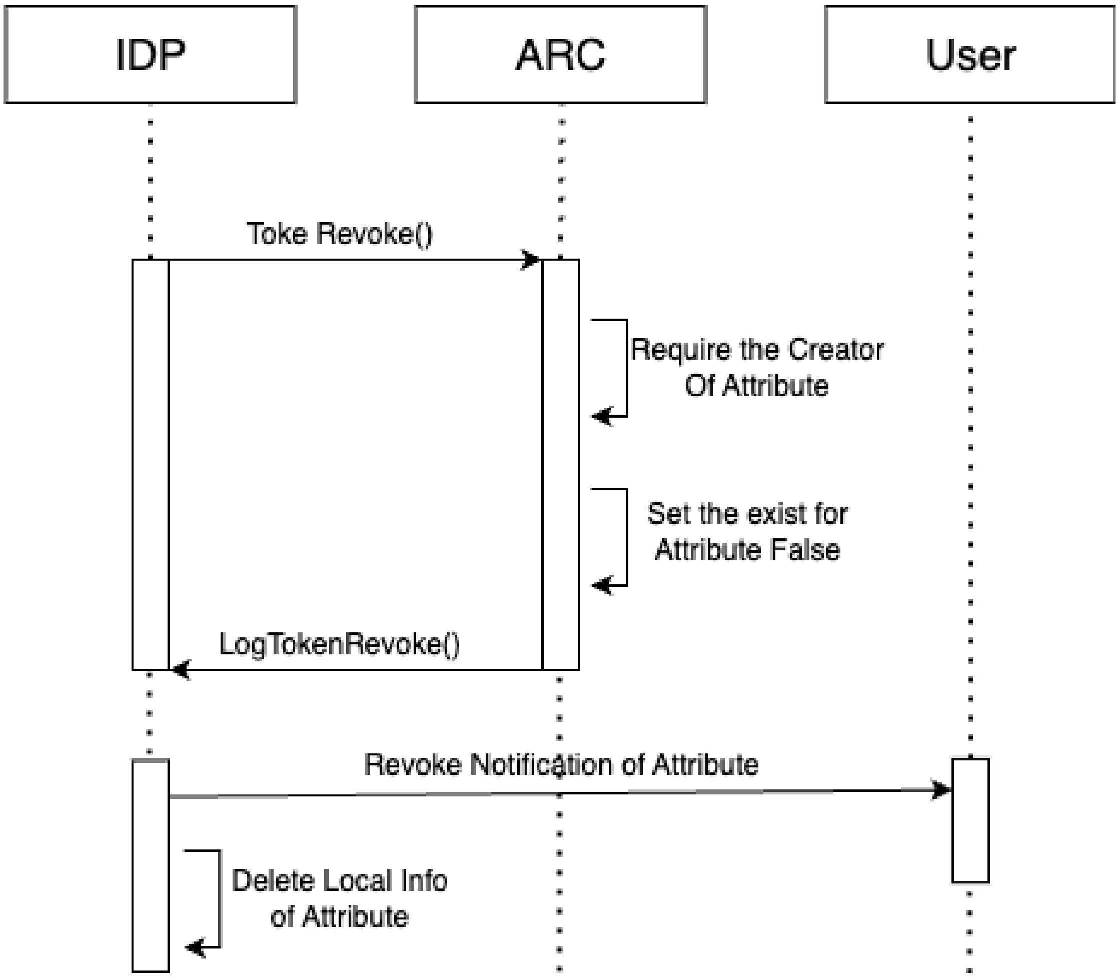
IDP
ARC
User
Toke Revoke()
Require the Creator Of Attribute
Set the exist for Attribute False
LogTokenRevoke()
Revoke Notification of Attribute
Delete Local Info of Attribute

Figure 7.7: Sequence diagram of revocation.

Algorithm 4 ARC.tokenRevoke()

```
1: **Inputs:** $\tau$
2: **Outputs:** LogTokenRevoke()
3: Require ARC.creatorQuery($\tau$) == msg.sender
4: Set the 'exist' of $\tau$ to False
5: Emit the event LogTokenRevoke()
```

After watching the event LogTokenRevoke(), the IdP notifies the user of the revocation of τ. Then both parties delete the local attribute information.

7.5 Challenge-response protocol:

The challenge-response protocol allows the user u to demonstrate to the SP that he/she possesses the identity identifier and attributes asserted in the profile without revealing the ZK private key. However, the sender of the transaction will change the status of the attribute, leading to the exposure of the association between the address and token. Therefore, u should use the one-off anonymous addresses in this protocol.

Before launching the challenge-response protocol, u already knew and possessed the required set of attributes for accessing the service. The sequence diagram of the challenge-response protocol is shown in Figure X. And the five components of the protocol are described as follows:

1. **Assertion:** u sends an assertion of profile packaged in JSON format to the SP through the secure channel. Then SP retrieves and confirms the name-value content and the authority of the creator of each privacy attribute token in ARC.

2. **Challenge:** For each attribute asserted in the profile, mapping with a 256-bit challenge factor is generated. Then the SP sends these mappings to u through the secure channel and starts watching `LogTokenResponse()` for these attributes.

3. **Response:** The user needs to compute responding content for each received challenge factor. We assume the challenge factor mapping to τ is π. Next, u uses the blockchain private key $(sk^E)_u$ of address E_u to sign π to get the signature S_π:

   $$\text{signatureGen}(\pi, (sk^E)_u) \rightarrow S_\pi \quad \ldots (1)$$

   Then u uses the SP's blockchain public key $(pk^E)_{\text{sp}}$ to encrypt the signature S_π to get the responding content ES_π:

   $$\text{contentEnc}(S_\pi, (pk^E)_{\text{sp}}) \rightarrow ES_\pi \quad \ldots (2)$$

   Then u needs to repeat the procedure of responding (elaborated above) anonymously to trigger `LogTokenResponse()` with each responding content.

4. **Validation:** The SP watches `LogTokenResponse()` via the event filter and verifies each triggered responding content according to the mappings of factors. The SP decrypts the responding content contained in `LogTokenResponse()`:

   $$\text{contentDec}(ES_\pi, (sk^E)_{\text{sp}}) \rightarrow S'_\pi \quad \ldots (3)$$

   The SP confirms that u signed the factor with the blockchain private key $(sk^E)_u$, where verificationResult $\in \{\text{True}, \text{False}\}$:

   $$\text{signatureVerify}(\pi, S'_\pi, (pk^E)_u) \rightarrow \text{verificationResult} \quad \ldots (4)$$

5. **Authorization:** If all verification results are True, the SP performs an authorization operation to grant the access request from u.

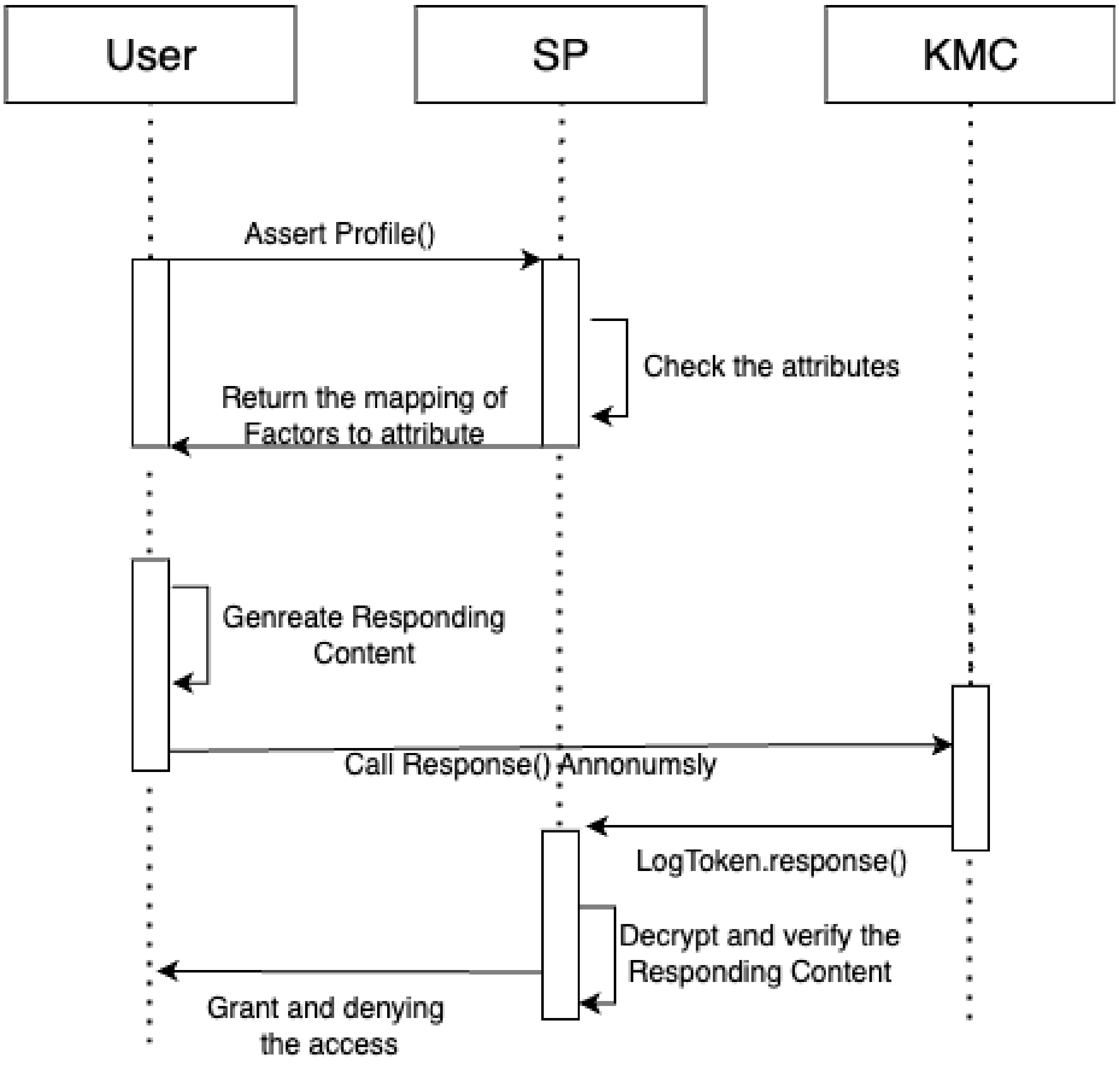
User
SP
KMC
Assert Profile()
Check the attributes
Return the mapping of Factors to attribute
Genreate Responding Content
Call Response() Annonumsly
LogToken.response()
Decrypt and verify the Responding Content
Grant and denying the access

Figure 7.8: Sequence diagram of the challenge-response protocol.

8 Security analysis:

The ideal DIMS in blockchain should guarantee the security of the bottom layer and the application layer. At the bottom, all transactions should be executed correctly. At the application layer, the user's privacy cannot be leaked.

The preconditions required for security analysis of the proposed scheme are as follows:

1. The SP will not disclose the attribute ownership information asserted in the profile.
2. The blockchain private keys and ZK private keys are properly kept and not leaked.
3. All preimage information stored in the local database is properly kept and not leaked.

8.1 The security of the bottom layer:

- **Zero-knowledge security:** Zokrates is based on the zk-SNARK algorithm groth17, and its security has been proven in the original literature.
- **Particular computation Security:** The formulas are introduced to analyze the security. For, how to guarantee the correctness of the private inputs and how to prevent leakage of user privacy when the public inputs are publicly verified.

- **Prevention of the replay attack:** The attacker obtains the partial inputs $\{\sigma_{\text{response}}, \text{pubInp}_{\text{response}}\}$ of `tokenResponse()` entered by the legitimate user from transactions. The attacker asserts to SP that he/she owns the privacy attribute token τ and receives a challenge factor θ. Then the attacker signs θ with his/her blockchain private key and uses the SP's public key to generate a responding content ES_{θ}, which combined with $\{\sigma_{\text{response}}, \text{pubInp}_{\text{response}}\}$ will serve as the attack resource. Then RVC verifies the proof and the corresponding public inputs according to the formula:
  `RVC.responseSKVerify(`$\sigma_{\texttt{response}}$`, pubInp`$_{\texttt{response}}$`)`.
  The returned result is `True`, but according to Algorithm 3, `token.Response()` detects that the waste $\text{pubInp}_{\text{response}}$ is in revocationList of KMC, failing to emit `LogTokenResponse()`. Therefore, this scheme can prevent a replay attack.

8.2 The security of the application layer:

- **Data minimization:** When accessing services from the IdP, the user needs to follow the challenge-response protocol. The profile that contains the necessary attributes to access the service is asserted, instead of all user attributes. Besides, the SP cannot derive all the attributes held by the user based on the user's blockchain address.

- **The unlinkability of identity:** If the attacker wants to know the owner of the privacy attribute token, he/she needs to crack the ZK public key contained in the hashclaim. According to preconditions 1 and 3, if the SP is honest and the preimage information stored in the database is not leaked, the attacker can obtain the pk_u^Z and ε_{τ} only through brute force cracking against the formula:

$$H_\tau = \mathrm{sha256}(\tau || \mathrm{pk}_u^Z || \varepsilon_\tau) \tag{8.1}$$

The 256-bit H_τ is the sha256 hash of 256-bit τ, 128-bit pk_u^Z, and 128-bit ε_τ. Among them, τ is public input, so the attacker needs to crack a remaining 256-bit preimage. However, using the current computing power to calculate the hash 2^{256} times is considered impossible. Therefore, this solution can realize the unlinkability of the user's identity.

- **Protection of behavior privacy of the user:** When the user u performs the challenge-response protocol legally, the used one-off anonymous addresses $\{u_1, u_2, ...\}$ are generated randomly and independently. The blockchain address E_u of u is independent of the anonymous addresses $\{u_1, u_2, ...\}$, thus the attacker cannot deduce the blockchain address that represents the user's real identity from the anonymous addresses calling tokenResponse(). After the event LogTokenResponse() is triggered, the responding content cannot be decrypted by the attacker since the responding content will not be linked to the blockchain address of any user. It can only be decrypted and verified by the SP. Others cannot even know which SP the user is requesting to from the protocol. Therefore, this solution can protect the behavior privacy of the user.

9 The related works:

The birth of blockchain technology has brought possibilities to solve digital identity puzzles. From industry to academia, all are trying to manage digital identity from a decentralized perspective.

In industry, the W3C Credentials Community Group designs DIDs (Decentralized Identifiers) to implement the DPKI (decentralized PKI) architecture. DIDs are a new type of identifier for verifiable, decentralized digital identities and do not rely on any centralized registry. Entities can create and manage their identifiers on any number of distributed, independent roots of trust. DID methods may also be developed for identifiers registered in federated or centralized DIMS.

Besides, companies such as Sovrin (2018) and uPort are committed to the application implementations of DIMS. Sovrin is a protocol in the distributed network for self-sovereign identity and decentralized trust, achieving a complete stack to manage identity from account to entity. Users can issue verifiable certificates containing a digital signature and selectively expose their identity through zero-knowledge proofs. However, in the consensus mechanism PBFT, the existence of stewards may spawn a group of supernodes acted by large companies, which runs counter to decentralization. uPort is a secure and self-controlled DIMS that allows users to act as their own CA. The agent contract address, as the core of uPort, acts as the globally unique persistent identifier. Users can submit verifiable proofs for applications, devices, or services and revoke them if necessary.

In academia, the SCPKI implements the idea of The Trust Web through smart contracts, building an example of the claim identity model. Any entity could act as a registrar to sign other people's attributes or revoke the signature. The SCPKI can easily detect rogue certificates when they are published, but a globally transparent claim model may pose a threat to identity privacy. Azouvi et al. (2017) designed a blind register protocol on the blockchain, allowing users to publish blind attributes in the blockchain, and IdP can sign on blind attributes. Users who have collected blind signatures can unblind them locally. This protocol addresses the question of how to register identities and attributes in a system built on globally visible ledgers. The scheme designed by establishes reputation-based DIMS on the blockchain and ensures the reliability of identity via smart contracts. To solve the problem of identity and access control in the permissioned blockchain, ChainAchor (Hardjono and Pentland, 2019) provides an anonymous but verifiable identity for entities. The identity is verified by a verification node built on tamper-resistant hardware that forms a privacy protection layer, allowing the entity to selectively disclose the identity during transactions. A new transitively closed undirected graph authentication (TCUGA) aims to bind digital identity to the real-world entity. The TCUGA allows the entity to verify membership via signatures, allowing dynamic addition or removal of vertices and edges in undirected graphs to update existing certificate relationships. Also, the self-sovereign identity scheme proposed by Stokkink and Pouwelse (2018) is adopted by the government, which created a new Dutch digital passport that could prevent identity fraud and provide legal force among citizens. The claim could be verified within a sub-second time.

10 Conclusion

This paper introduces zk-SNARK into the existing claim identity model to safeguard identity privacy. It proposes a method for secretly transferring ownership of privacy attributes and authenticating attribute ownership through the use of privacy attribute tokens and specific computations. This framework adopts off-chain computing and on-chain verification, effectively preventing the exposure of ownership between user entities and attributes in the distributed ledger, thus achieving identity unlinkability and behavior privacy. Experimental evaluation and analysis demonstrate that the designed challenge-response protocol enables low-cost and high-throughput authentication processes, applicable to security mechanisms such as access control and authorization. Future work will focus on addressing the shortcomings of Schema to broaden its applicability. For instance, the redundancy of attributes with the same key-value pair created by IdP for different users needs to be minimized to reduce operating costs. Additionally, efforts to reduce the time required for off-chain proof generation, perhaps by implementing a custom ZKP library instead of relying on Zokrates, will be explored.

11 References

1. https://blockchain.gov.in/Home/BlockChain?blockchain=feature
2. https://blockchain.gov.in/Home/BlockChain?blockchain=feature
3. https://www.youtube.com/watch?v=RkYVVC2vXho
4. A Brief Dive Into zk-SNARKs and the ZoKrates Toolbox on the Ethereum Blockchain
5. zkSNARKs in a Nutshell Christian Reitwießner chris@ethereum.org
6. https://www.sciencedirect.com/topics/computer-science/smart-contract
7. Blockchain-enabled decentralized identity management: The case of self-sovereign identity in public transportation
8. http://library.usc.edu.ph/ACM/SIGSAC
9. A systematic literature mapping on secure identity management using blockchain technology
10. https://www.gep.com/blog/mind/guide-to-smart-contracts-for-supply-chain
11. https://consensys.io/blockchain-use-cases/digital-identity
12. https://www.sciencedirect.com/science/article/pii/B9780128038437000715/pd
13. https://www.itu.int/dms_pub/itu-t/oth/06/04/T06040040040001PDFE.pdf

14. https://www.okta.com/resources/whitepaper/practical-thoughts-on-blockchain-and-identity/

15. https://widgets.weforum.org/blockchain-toolkit/digital-identity/index.html

16. https://www.sciencedirect.com/science/article/pii/S1319157821000690

17. https://medium.com/@pavelkravchenko/ok-i-need-a-blockchain-but-which-one-ca75c1e2100

18. https://medium.com/cornellblockchain/a-brief-dive-into-zk-snarks-and-the-zokrates-toolbox-on-the-ethereum-blockchain-cb7bd7f00fdc

19. https://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf

20. https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell

21. https://encrypted-tbn0.gstatic.com/imagesq=tbn:ANd9GcT6J_-7GO7I4M9ck_0PeEpQ1jV5V1YdbBcTbxgs