Web3-Powered service provisioning in cellular networks using NFT and self-sovereign identity

Nischal Aryal, Fariba Ghaffari, Emmanuel Bertin, Noel Crespi

Web3-Powered Service Provisioning in Cellular Networks using NFT and Self-Sovereign Identity

[1, 2] Nischal Aryal, [2] Fariba Ghaffari, [1, 2] Emmanuel Bertin, [2] Noel Crespi
[1] *Orange Innovation, 14000 Caen, France*
[2] *SAMOVAR, Telecom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France*
{nischal.aryal, emmanuel.bertin}@orange.com, fariba.ghaffari@telecom-sudparis.eu, and noel.crespi@it-sudparis.eu

***Abstract*—The rise of internet and data usage highlights the importance of service provisioning for Mobile Network Operators (MNOs) in expanding their operations and meeting user demands. Implementing scalable and secure authentication and access control mechanisms is crucial for enabling service utilization among eligible users and ensuring the viability of emerging business models. Conventional centralized approaches face limitations such as single-point-of-failure, low scalability, computational overhead, and privacy vulnerabilities. MNOs must explore innovative business models to augment revenue streams and address these challenges. Realizing such models involves automating user contracts with service providers and safeguarding user privacy regarding data sharing with external entities. Blockchain technology offers a transformative avenue for integration within existing MNO infrastructures, providing novel distributed authentication and access control methodologies. We propose a new business model for MNOs and service providers in which an Attribute-Based Access Control (ABAC) framework handles user access to services on top of Blockchain. Moreover, a tokenized data-sharing method facilitates selective data sharing with service providers through MNO channels based on user-defined permissions. Central to this approach are non-fungible tokens (NFTs) and the Self-Sovereign Identity (SSI) paradigm, where NFTs ensure secure, decentralized tokenization of user data, and SSI empowers users with ownership and control over their data. Assessments confirm the scalability of this solution, making it suitable for different use-case requirements.**

***Index Terms*—NFT, Cellular Network, SSI, DLT, Blockchain, Smart contract**

I. Introduction

There is an exponential growth in data consumption due to the widespread adoption of smartphones, diverse mobile applications, IoT devices, and emerging technologies, such as AI/ML [1], [2]. In this landscape, Mobile Network Operators (MNOs) must take some strategic actions to ensure customer satisfaction, revenue growth, and a competitive advantage. For this, service provisioning is critical because it enables the efficient delivery of a wide range of services, including content streaming, messaging, video conferencing, and gaming, allowing MNOs to meet customer expectations. Proper service provisioning can be accomplished through optimizing network design, managing resources, establishing service agreements, and managing the Quality of Service.

In a simple service provisioning scenario, users first subscribe to a service through an online channel, sharing their Personally Identifiable Information (PII). The Service Providers (SPs) validate and store the users' information via a central authority and provide the users with access to the service. There are several challenges with this scenario concerning technicalities and data privacy. From a technical standpoint, the central authority responsible for subscription and access control is a single point of failure. It also leads to limited scalability, IT complexity, lack of automation, and low fault tolerance. From a data privacy standpoint, users often reveal PII across multiple online platforms without full awareness of the repercussions. When users share their PII to different platforms, the platforms store their information on private servers, which could lead to malicious activities. Once the users share the information, they have little control over how the platform uses their data. Any solution that can provide a distributed management strategy with greater flexibility, automation, and robustness while maintaining the ability of data owners to control how their data gets used would be considered a promising candidate to address these concerns.

Blockchain as a Distributed Ledger Technology (DLT) is a cryptographically secure network of nodes in which all changes in the system require the consensus of all the eligible nodes. Thanks to its unique features, such as immutability, transparency, and consensus mechanisms, there is a research interest in using Blockchain for identity-related aspects such as authentication, access control, identity management, and data or resource sharing. Along with these features, Blockchain also provides some intriguing security and privacy use cases, such as Non-Fungible Tokens (NFTs) and Self-Sovereign Identity (SSI). NFTs are unique digital assets that cannot be duplicated, divided, or traded. They allow the efficient and secure verification and ownership management of digital assets and offer solutions to secure them [3]. Self-Sovereign Identity (SSI) gives control to data owners over their data by allowing them to define policies to access them [4]. In SSI, data owners can independently present identity claims to the data requester and verify the claims with cryptographic certainty through Blockchain.

In this paper, we propose a Blockchain-based service provisioning model using NFT and SSI to address the aforementioned challenges. This method implements authentication and access control procedures based on smart contracts. We implemented a fine-grained Attribute-Based Access Control (ABAC) technique on top of the Blockchain, where SPs can validate the ownership of the NFT by matching the user's signature on the access request transaction and the MNO's

signature on the NFT. Using unique NFTs for each user eliminates the need for intermediaries during user authentication. Additionally, SSI incorporates the "Zero-Knowledge Proof" concept, where the SP can authenticate the user without any knowledge of their PII or the data within the NFT. This method helps service providers verify the user's identity without having access to the user's sensitive data.

The key contributions of our proposed method are as follows:

- Eliminating the need for intermediaries during service-based subscription and access control procedures,
- Providing flexible and automated subscription and access control process based on a zero-knowledge proof, and
- Giving data owners complete ownership over their data.

The rest of this paper is organized as follows: Section II provides a brief background, followed by a summary of the state of the art in Section III. Section IV outlines the system design and construction of our proposed method, followed by the evaluation in Section V. Section VI provides our conclusions about the proposed method as well as some future research directions.

II. Background

This section presents the fundamental background required for the rest of the paper including access control, NFTa, and SSI concepts.

- **Access Control** ensures that only authorized entities can access the resources. This can be accomplished using a variety of techniques. In this paper, we utilized the ABAC solution, which is a fine-grained method supporting different constraints to define the legitimate user and a context-specific solution to define the access policies by the resource owners. ABAC has four sets of attributes. *Subject* attributes specify the subject, *Object* attributes distinguish the resources, *Action* attributes are the actions that can be performed by the subject, and *Environment* attributes describe the context in which access is requested.
- **Self-Sovereign Identity (SSI)** is a digital identity model where an individual has full control over their data [5]. It enables individuals to securely store and manage their personal information without relying on intermediaries. Moreover, they can selectively disclose their personal information in various scenarios, such as interacting with businesses, using online services, etc. Note that, in the SSI model, the users can store their data locally, in an external database, or within the databases of claim issuers [6], [7].
- **Non-fungible Tokens (NFTs)** are digital assets that prove ownership of an individual item or information and are usually stored on Blockchain. This concept, first implemented by Ethereum's $ERC-721$ token standard and developed in $EIP-1155$ [8], is a unique digital asset that cannot be duplicated or divided. Unlike cryptocurrencies, which have identical characteristics and can be traded one-to-one (fungible), NFTs are indivisible and cannot be exchanged due to their distinct properties and metadata (non-fungible) [9].

III. Related Works

Due to the significance of access control in providing services for the users, and for preserving the user's privacy, a variety of solutions are proposed in this area. A considerable part of Blockchain-based access control solutions in service provisioning literature are dedicated to centralized systems with a trusted central authority to manage user access. Despite the low implementation complexity of these methods, they suffer from a single point of failure, low scalability, low availability, low non-repudiation, and lack of proper privacy preservation [10], [11].

Based on our survey regarding the existing state of the art in access control solutions, this technology is mainly used for (1) defining and storing the access policies and rules in smart contracts [12]–[17], (2) verifying the user's access request [13], [14], [16], [18], and (3) enforcing the access control decision ([13], [16]).

Along with access control, some solutions offer privacy for data sharing in different sectors, such as healthcare and finance, using SSI. For instance, Zhuang et. al [19] propose a Blockchain-based access control system for sharing the patient's PII data in the healthcare sector with the doctors through the SSI in which the user decides about the data which they want to share. Ahmed et al. [20] propose a Blockchain-based SSI system focusing on the financial sector.

While many methods have been proposed for providing access for users in different sectors, only a limited number of these studies provide self-sovereign management of the user's identity and address user privacy. Moreover, to the best of our knowledge, a very limited number of works focus on service provision, access control, and privacy preservation in the cellular network sector.

IV. System overview

In this section, we explain in detail how our proposed smart contracts work and how it deals with three main tasks: 1) **NFT generation**: allows users to create and save NFTs in their smart contracts, 2) **Service-based registration**: allows users to sign up for services and service providers to set rules for user access, and 3) **Authentication and Access Control**: verifies if users are who they say they are and decide if they can use services. Fig. 1 shows an overview of how the method works.

Note that to design this system, we assume that the following steps are performed:

- The user has a subscription with the MNO, meaning they have internet access via their SIM card's mobile data or other types of internet connection provided by the MNO.
- During the subscription procedure, a user has provided their PII to the MNO (i.e., same as the current subscription process). So, it is assumed that the MNO possesses the user's PII data, including their name, surname,

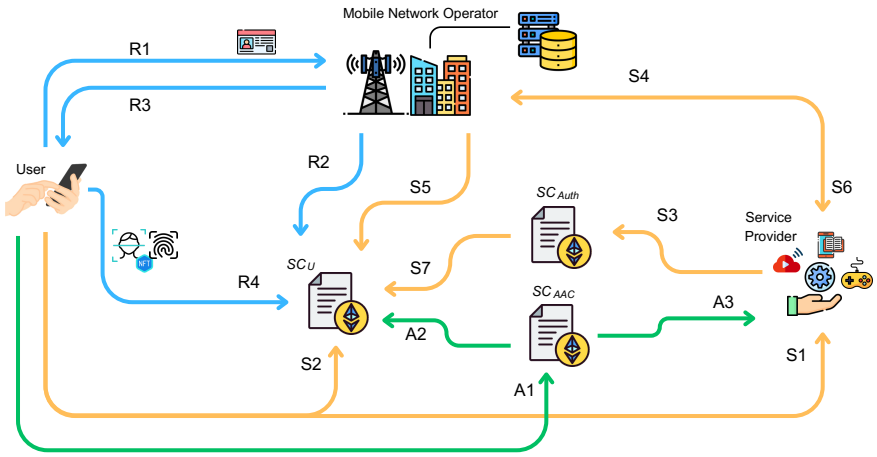


Fig. 1. An overview of the proposed design. R1-R3 (blue lines) show the token generation steps, S1-S7 (yellow lines) show the service-based registration process, and A1-A3 (green lines) summarize the authentication and access control processes. For simplification, only the core smart contracts are mentioned in the figure. The remaining smart contracts are described in Section IV-A

address, identity photo or photo of identity card, bank account information, and so on.

- Before users can subscribe to services, SP must register with the system and agree with the MNO. This implies that the MNO and the service provider have already agreed on offering specific services with certain benefits to users through the MNO's platform. The details of this procedure are beyond the scope of this paper. However, we have addressed a similar process in our earlier work [21].

Given these assumptions, the initial phase involves *generating* $NFTs$, denoted as R1 to R4 in Fig. 1. In this step, the user initiates a subscription request for the Blockchain-based service. Initially, the MNO establishes a dedicated smart contract, denoted as SC_U, for the user. Subsequently, the user accesses the application using credentials provided by the MNO post-subscription. Next, the user configures their biometric, facial recognition data, or any other authentication method specific to their device to generate an NFT, sends it to the MNO for validation and signing, and finally, stores it within their contract application.

Following NFT generation, *users subscribe to various services* offered via the Decentralized Application (DApp), denoted as S1 to S7 in Fig. 1. Once a user selects a provider, the provider's Blockchain address or specific code is included in a list of approved services within the user's contract. This allows users to submit their NFT for initial registration, and if further data is required by the service provider, users can grant or deny access through their smart contract. Leveraging this access mechanism, service providers can request the MNO to share specific data. Upon successful registration, the user's access policy, comprising the service plan, pricing, expiration date, and other access attributes, is appended to their contract.

After completion of the service-based subscription, users can request *access to the services* (referred to as A1 to A3 in Fig. 1) by transmitting their wallet addresses to the authentication and access control contract. Subsequently, the authentication and access control contract cross-references the user's access attributes with the policies stored within their smart contract. Based on this evaluation, the contract either grants access permission or rejects the access request.

A. Proposed Smart Contracts

1) ***Address Book Contract*** *($SC_{AddBook}$):* Address Book is a unique and tamper-proof contract that stores the addresses of all other single contracts, such as SC_{DB}, SC_{EP}, SC_{Reg}, and SC_{AC} that are required for token generation, service-based subscription and authentication and access control processes. We designed this contract to avoid the use of hard-coded addresses and to handle maintainability defects of smart contracts [22]. Moreover, having a list of addresses of the other contracts allows managing and implementing the modifiers in functions to benefit from the intrinsic access control capability of smart contracts, and to provide more secure collaboration among them.

2) ***Eligible Providers Contract*** *(SC_{EP}):* This contract records the addresses of all eligible service providers in the Blockchain-based service provisioning system. If a service provider can reach an agreement of service with the MNO, their Blockchain address will be added to this contract. Note that, as mentioned before, the agreement and negotiation process between an MNO and a service provider is supposed to be off-chain, and in this paper, we will not focus on this phase. The purpose of designing this contract is that SC_{Reg} and SC_{AC} can verify the eligibility of the providers to whom the users are demanding to connect.

3) ***User Address Database Contract*** *(SC_{DB}):* This contract contains a list of user-specific contract addresses SC_U and the user's wallet address, $Wallet\_Address_U$. When a user subscribes to the Blockchain-based services provided by an MNO and the MNO deploys a smart contract for them, the record will be added for that specific user to map the $Wallet\_Address_U$ to the SC_U.

*4) **User Contract** (SC_U):* This is a user-specific contract that stores the information required for further authentication and access control procedures. The SC_U contract is created and deployed by the MNO after verifying the user subscription request to the Blockchain-based service provisioning application. The user's NFT, the list of their registered/subscribed services, the balance of their wallet, and the permission vector are some of the most important data stored in this contract. The permission vector is a vector of bits that shows in which personal data, the user permits MNO to share with the service provider.

*5) **Registration Contract** (SC_{Reg}):* This contract is responsible for handling service-based registration requests by verifying the identities of the user and service provider and, then, inserting the required policies in the user's smart contract.

*6) **Access Control Contract** (SC_{AC}):* This contract is responsible for controlling the user's access to the services by validating the user's current attributes with existing policies in their smart contract.

B. System design

We detail the three main functionalities of the proposed method. First, the following setup must be done in the system:

- The $SC_{AddBook}$ contract needs to be deployed. To do so, the system admin (i.e., one of the addresses registered on behalf of the MNO) deploys it. Whenever a smart contract searches for a particular smart contract, it can refer to the $SC_{AddBook}$ contract.
- Next, the admin deploys the SC_{EP} and SC_{DB} contracts and stores their addresses in $SC_{AddBook}$.
- The addresses for all eligible service providers are also stored in the SC_{EP} by the MNO. We assume that the system admin checks the eligibility of all service providers before adding them to the SC_{EP} contract (i.e., through an off-chain agreement).
- As previously discussed, users in the SSI model have the options to store their data in various locations such as local storage, external databases, or within the databases of the claim issuer. In this study, we assume that users store their data on the claim issuer's site and manage access permissions to this data using Blockchain technology [6], [7].

*1) **Token generation**:* The main purpose of this step is to generate a unique NFT token for the user and store it securely in the Blockchain. This generation requires the following steps (the enumeration is based on Fig. 1):

(R1): The user logs in to the DApp developed by the MNO and enters their credentials. Note that, since the MNO develops the DApp, the user can log in with a unique, one-time secure link or other predefined credentials such as the user's biometrics (e.g., fingerprint or face ID), secure password, etc. Given this information, the user requests the MNO for permission to use the provided Blockchain-based service.

(R2): Once MNO receives the credential and required PII of the user through a secure channel (e.g., the DApp), it matches the user's data with stored PII. If the verification is successful, MNO deploys a specific smart contract, SC_U, for the user in Blockchain.

(R3): Along with the $Address_{SC_U}$, MNO sends a secure, on-time, unique link for the user to generate the NFT. By opening the link, DApp can request that the user enter their fingerprint, face ID, dedicated phone PIN, or other unique data, as well as an SMS code for two-factor authentication. Using these data, DApp will parse the generated strings from two different factors and calculate their hash to generate a unique NFT for the user. The NFT is sent to the MNO through the secure channel provided by DApp. MNO signs the NFT using its private key and sends it back to the user.

(R4): Once the user receives the signed NFT, they can create a transaction to send the NFT to the SC_U using the $Wallet\_Address_U$. The SC_U verifies the sender of the message and if `msg.sender == owner`, it will register the user's NFT.

*2) **Subscription in service**:* In this step, the user, whose NFT is stored in the SC_U, registers for the services provided by the legitimate service providers and accepts/denies the provider's access to their PII data stored in the MNO. The following steps are needed for the user registration (the enumeration is based on Fig. 1):

(S1): The user opens the specific tab of the desired service provider in the DApp and selects the suitable plan. The user will then be directed to the page to accept/deny the data that the service provider asks the user to share. Note that, in this paper, the main data that the user needs to share is the NFT. The other data, based on the user's preference, incentives of the service provider, etc. can be allowed or denied for further sharing.

(S2): The user's selected options are fed into a `permission_vector` as a list of bits, which is set to 1 if the user accepts sharing the data, and 0 when they deny sharing. This transaction is sent to the SC_U with $Wallet\_Address_U$ as the sender. The SC_U checks the transaction's sender, and if `msg.sender == owner`, it will register the user's `permission_vector` for the specific service provider. Note that the user can give minimum access in this step and then increase permissions in the future. When these steps are finished, DAPP informs the service provider.

(S3): The service provider calls the `validation and registration()` function of the SC_{Reg} to validate the user's information and authenticate the user. First, the SC_{Reg} fetches the $Address_{SC_{EP}}$ and $Address_{SC_{DB}}$ contracts from the $SC_{AddBook}$. Once the addresses of these two contracts are available, SC_{Reg} first checks in the SC_{EP} to determine if the service provider is eligible to send requests. If that eligibility is confirmed, SC_{Reg} sends a request to SC_{DB} to get the address of the SC_U using the $Wallet\_Address_U$ sent by the service provider. After receiving the address, SC_{Reg} sends an event to

the DApp requesting the biometric information of the user. When the user receives a notification to provide their biometric data, another transaction containing this information is sent to SC_{Reg}. Upon receiving this data as an array of bytes from the DAPP, SC_{Reg} redirects to SC_U to retrieve the user's valid NFT. SC_{Reg} then compares the received NFT (i.e., the array of bytes from the DAPP) with the user's NFT stored in SC_U. If these two values match, the user is authenticated.

(S4): The SC_{Reg} responds to the service provider that the user with an identification pair ($Address_{SC_U}$, $Wallet\_Address_U$) is authenticated. Using these data, the service provider sends the data-sharing request to the MNO.

(S5): When the MNO receives the request, it fetches the `permission_vector` recorded by the user in the SC_U. Note that, since the user has already sent a transaction in Blockchain to store this `permission_vector`, and the user validation is verified before recording the permission, we are assured that the only user could be the one to give this access to the service provider.

(S6): Based on the retrieved `permission_vector`, the MNO shares the information with the service provider for the next usage. Note that these data are not required for the user's next authentication for registration, access control, or payment, but can be used for further advertisement, providing content, etc.

(S7): In parallel with the last three steps (S4 to S6), the SC_{Reg} sends a request to the SC_U to add the new service to the user's service list by calling the `set_New_service()` function. First, the SC_U fetches the $Address_{SC_{Auth}}$ from $SC_{AddBook}$ and verifies the sender of the transaction. If `msg.sender` == $Address_{SC_{Auth}}$, the service will be recorded to the user's registered services.

*3) **Service-based Authentication and Access Control**:* In this step, the registered user requests a connection to the service of the service provider through the DApp. The Attribute-based Access Control (ABAC) solution is used in this phase. The main steps of the access control procedure are described below (the enumeration is based on Fig. 1):

(A1): The user sends a transaction to the SC_{AC} requesting a connection to a legitimate service. In this request, the provider's address or service code can be transmitted to the SC_{AC}, and the transaction is sent by the $Wallet\_Address_U$ along with the NFT.

(A2): After receiving the access request, SC_{AC} fetches the $Address_{SC_{EP}}$ and $Address_{SC_{DB}}$ contracts from $SC_{AddBook}$. Once the addresses of these two contracts are available, SC_{AC} first verifies that the service provider is on the list of eligible addresses in SC_{EP}. Once the verification is successful, SC_{AC} sends a request to SC_{DB} to get the address of SC_U using the $Wallet\_Address_U$ that is the $msg.sender$. By fetching the SC_U, SC_{AC} retrieves the policy and the access attributes to the desired service from the contract.

(A3): To verify the user's access eligibility to the service, assume that, in the policies, there are the following four attributes: 1) a Subject (SA), who is the user, 2) an Object (OA) that is the specific service, 3) an Environment (EA) which is the expiration time and the user's balance in their wallet, and 4) an Action attribute, which can be allowed or denied. SC_{AC} does the following verification through attributes:

$$SA = \begin{cases} 1, & \text{if } msg.sender = WalletAddress_U, \\ & Token' = Token \\ 0, & \text{otherwise} \end{cases}$$

$$OA = \begin{cases} 1, & \text{if } eligible(SP_{tx}) = True, \\ 0, & \text{otherwise} \end{cases}$$

$$EA = \begin{cases} 1, & \text{if } Balance_u \geqslant Price_{service}, \\ & CurrentTime > Expiration\_Time \\ 0, & \text{otherwise} \end{cases}$$

where $Token'$ is the token sent by the user and $eligible(SP_{tx})$ is the function of showing if the service provider that is mentioned in the transaction, is one of the eligible providers. Finally, the decision would be sent to the user and the service provider (as an event) through action attribute (AA) as follows:

$$AA = \begin{cases} 1(allow), & \text{if } SA = OA = EA = 1, \\ 0(deny), & \text{otherwise} \end{cases}$$

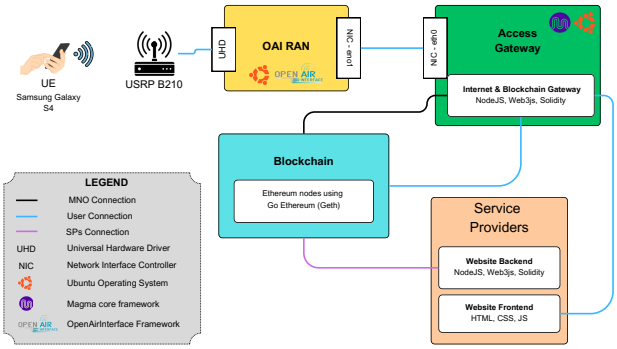
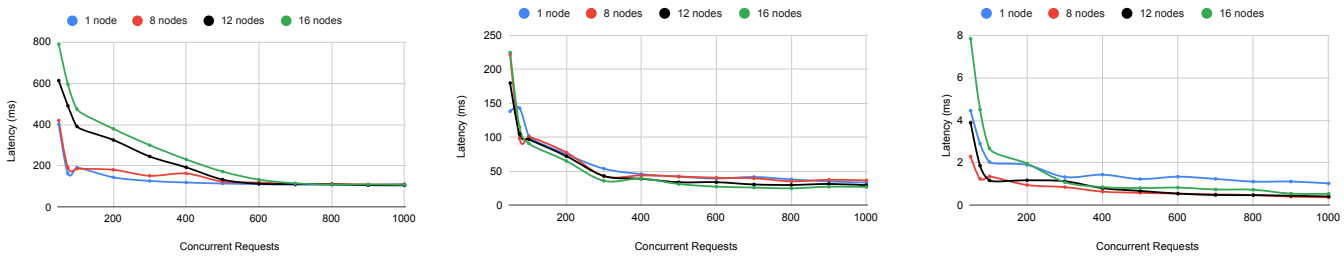


Fig. 2. A design overview of our testbed.

V. Evaluation

To evaluate our proposed method, we deployed the required testbed containing the MNO, the service providers, the users, and the Blockchain which all entities can connect to. To simulate the users and the MNO, we first deployed a private cellular network [23] using OpenAirInterface (OAI) and Magma core. OAI and Magma core are open-source frameworks for deploying cellular network components on commercially available hardware, such as UE, Radio Access



(a) (b) (c)

Fig. 3. Service-based Registration Latency

Network, and core network/access gateway. Next, we set up a private Ethereum network using the Go Ethereum (Geth) framework [24], [25]. Geth is a popular implementation of the Ethereum network developed in the Go language. The first step to deploy the private Blockchain is to create a genesis block (Fig. 4 depicts part of the genesis block in our setup), in which the $chainID$ can be a random number different than main-net or other test-nets, and consensus algorithm that, in our testbed, is *"clique"* [26], one of the Proof of Authority (PoA) consensus model's algorithms [27]. as the network's consensus protocol. PoA is a reputation-based method in which the reputation of the validators is the indicator of their selection. The validators in PoA have formally approved accounts, and their identity are public [11]. The smart contracts are developed using the Solidity [28] programming language. Finally, we set up a gateway for the testbed to communicate with the private Blockchain. In this configuration, the user first completes MNO-based registration (i.e., conventional 4/5G AKA procedure) before receiving internet service. When the user's equipment attempts to connect to our testbed, the access gateway executes the authentication and key agreement process, which validates the authenticity of the user's subscription to the internet service. Following successful authentication, the user gains internet access, through which he/she finally connects to the Dapp website provided by the service provider.

```
{
  "config": {
    "chainId": 769599,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "berlinBlock": 0,
    "clique": {
      "period": 5,
      "epoch": 30000
    }
  },
  "difficulty": "1",
  "gasLimit": "8000000"
```

Fig. 4. Configuration of genesis block in the system setup

The performance analysis of the suggested design is done in three parts: (1) Comparison with existing state of the art, (2) Evaluating the scalability in terms of increasing number of concurrent connection requests, and (3) Gas consumption of different on-chain processes.

A. Comparison with existing solutions

Table I compares the proposed method with other state-of-the-art methods for providing access control services and SSI solutions. To the best of our knowledge, we could not find a paper that shared the same scenario as this paper. Thus, we compare the more related state of the arts.

As shown in Table I, several works implemented the ABAC solution, while other works proposed different access control methods. It is important to mention that we only focused on comparing our work with papers that are more related to our use case. Using the SSI and tokenization of the authentication are subjects that are rarely provided by the other methods. These concepts can provide privacy and accountability in the system, which are two important factors in service provisioning through cellular networks. In Table I, compared to the works that use PBFT models, our proposed model can provide higher scalability, and in comparison to PoW, it is more efficient and provides higher performance regarding latency. Furthermore, several methods are using Blockchain for either storing policies or only validating the user's access to the system. Comparing our proposed method and these solutions, we can state that our proposed method can provide higher automation in authentication and access control by enforcing the access result for user's access to the services. This can remove any single point of failure in the access control procedure. So, we can claim that the proposed system is more fault-tolerant.

B. Scalability

We evaluate the scalability of our proposed method by analyzing the latency of on-chain processes -—token generation, service-based subscription, and service-based authentication and access Control– to handle different numbers of concurrent requests. The scalability of the system can be defined as changes in throughput or latency when altering one/several parameters. To measure scalability, we assess the latency using the following formula:

$$Latency = \frac{t_f - t_s}{|Tx|}$$

TABLE I
COMPARISON WITH EXISTING SOLUTIONS

Ref.	[12]	[15]	[18]	[14]	[13]	[20]	[19]	[16]	This work
Properties									
Access Control Automation	✓	✗	✗	✓	✓	✗	✗	✓	✓
Authentication automation	✗	✗	✗	✗	✓	✓	✓	✗	✓
Scalability regarding users	✗	✓	✗	✓	✓	✓	✓	✓	✓
Privacy-preserving	✗	✓	✓	✗	✓	✓	✓	✗	✓
Data sharing possibility	✗	✓	✓	✗	✗	✓	✓	✗	✓
Use of SSI	✗	✗	✗	✗	✗	✓	✓	✗	✓
Non-fungible identity	✗	✗	✗	✗	✗	✗	✓	✗	✓
Service provisioning in cellular network	✗	✗	✗	✓	✗	✗	✗	✗	✓
Blockchain role*	1,2	1	1	1,2,3	1,2,3	1,2	1,2	1,2	**1,2,3**
Access Control Model	ABAC	General	CP-ABE	ABAC	ABAC	-	-	ABAC	**ABAC**
Consensus model	PoA	PoS	PBFT	PoW	PBFT	PoA	Raft	PBFT	**PoA**

* (1) is using Blockchain as storage for policies, (2) is using Blockchain for access validation, and (3) means the method is using Blockchain for access enforcement.

Here, t_f represents the completion time of the simulation, while t_s denotes the starting time of the simulation. It's essential to clarify that in this context, simulation refers to the duration spanning from when concurrent requests are initiated to when transaction receipts for all requests are received.

In this evaluation, we altered the number of full nodes in the system up to 16 and the number of concurrent requests up to 1000. The results are depicted in Figure 3(a-c), which show that the latency of both processes is almost stable after 400 concurrent requests. Moreover, these figures also show that scalability is not affected when we increase the number of Blockchain full nodes. Note that each node in the Blockchain represents either an MNO or a service provider. It is also important to mention that the depicted latency in these figures is not representative of the user's experienced latency, it shows the average latency and the capacity of Blockchain to handle the number of concurrent requests.

C. GAS consumption

This section evaluates the GAS consumption of different function calls and contract deployments. The GAS is the fee that the sender must pay to submit transactions to the Ethereum network. The cost that is mentioned in this part is the cost of sending a transaction of a contract to the Ethereum blockchain (i.e., transaction cost) [29]. The GAS cost is defined in $Gwei$ (i.e., as $10^{(-9)}ETH$). Table II shows the GAS cost in different processes. It is important to mention that in private or consortium Blockchains, the price of sending or processing the transaction is based on the agreement among all participating entities, and no currency is mandatory [11].

TABLE II
GAS PRICE OF PROCESSES AND TRANSACTIONS OF THE PROPOSED METHOD

Process	Transactions	Tx cost
Contract deployment	$SC_{AddBook}$	266550
	SC_{DB}	191518
	SC_{EP}	398664
	SC_U	549967
	SC_{AC}	590228
	SC_{Reg}	666643
Provider registration	Add new eligible provider	77329
User registration	Add SC_U in SC_{DB}	50975
User registration	Add new eligible provider	77329
	Authenticate and add policy	115957
Subscription	NFT validation and User subscription	0*
Access control	Access verification using saved policies	0**

* In this step, the user's biometric and address are sent to the smart contract for verification. Through the process, nothing needs to be saved in a smart contract, and the ledger's state doesn't need to change. So, the cost is zero.
** This process is a `call` not `transaction`. So, since it doesn't change the state of the ledger, the cost is zero.

VI. DISCUSSION AND FUTURE DIRECTIONS

In this paper, we present a system to offer services using Blockchain technology, potentially creating a fresh business model for mobile network operators and their partnerships with other service providers. We tackle the challenge of safeguarding user privacy and giving them full ownership and control over their data. To accomplish this, we utilize access control methods and authentication techniques like Non-Fungible Tokens (NFTs) and the Self-Sovereign Identity (SSI) approach. Our main aim is to develop flexible, automated, and easily scalable solutions for subscription-based services and user authentication and access control, ensuring that individuals maintain complete authority over their own data.

We put the system into operation by setting up a private cellular network with multiple simulated service providers utilizing the Ethereum Blockchain with a maximum of 16 full nodes. Evaluating the scalability of the system involved varying the number of full nodes up to 16 (which could represent either service provider or MNO nodes within the Blockchain) and simulating up to 1000 users with concurrent requests. Our findings show that even as the number of nodes

or requests increases, the latency remains largely consistent, revealing the high scalability of the system.

However, to put our proposed system into practice, several questions have to be addressed, such as selecting the type of blockchain, defining the role of different actors in the system, and establishing ownership of the Blockchain in real-world scenarios. Our system is designed to serve a particular group of stakeholders. So, we design a permissioned consortium Blockchain that is flexible based on the needs of different parties, and it only allows the participation of entities from the consortium. The key players in this system are the users, mobile network operators (MNOs), and service providers. Among these parties, users have limited processing power, storage, and resources, rendering them unable to function as full nodes within the Blockchain. Consequently, users in this setup act as light nodes, retaining only the portion of the Blockchain's information needed for conducting transactions while abstaining from involvement in the consensus process. Thus, the underlying Blockchain structure could be configured as a consortium of MNOs and service providers, determining transaction costs (if applicable), consensus models, storage methods, and related parameters.

Several areas could be explored in future research to enhance the system's performance and its practical application. For example, the current method for registering/subscribing providers and their agreements with MNOs to deliver services relies on manual agreements with pre-defined prices and Service Level Agreements (SLAs). Automating this process securely and in a more scalable manner could be achieved using Blockchain technology and smart contracts. Additionally, since Blockchain technology demands substantial storage in its full nodes to maintain an updated ledger and ensure security, enhancing storage efficiency is another aspect to consider. Implementing chain sharding could prove highly advantageous in addressing storage requirements [30].

REFERENCES

[1] "Ericsson Mobility Report," Nov. 2023. [Online]. Available: https://www.ericsson.com/4ae12c/assets/local/reports-papers/mobility-report/documents/2023/ericsson-mobility-report-november-2023.pdf

[2] N. Aryal, F. Ghaffari, E. Bertin, and N. Crespi, "Subscription management for beyond 5g and 6g cellular networks using blockchain technology," in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–7.

[3] B. Hammi, S. Zeadally, and A. J. Perez, "Non-fungible tokens: a review," *IEEE Internet of Things Magazine*, vol. 6, no. 1, pp. 46–50, 2023.

[4] "The path to self-sovereign identity," https://www.lifewithalacrity.com/article/the-path-to-self-sovereign-identity/, accessed: 2024-04-07.

[5] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Computer Science Review*, vol. 30, pp. 80–86, 2018.

[6] Š. Čučko and M. Turkanović, "Decentralized and self-sovereign identity: Systematic mapping study," *IEEE Access*, vol. 9, pp. 139 009–139 027, 2021.

[7] R. Nokhbeh Zaeem, K. C. Chang, T.-C. Huang, D. Liau, W. Song, A. Tyagi, M. Khalil, M. Lamison, S. Pandey, and K. S. Barber, "Blockchain-based self-sovereign identity: Survey, requirements, use-cases, and comparative study," in *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, ser. WI-IAT '21. New York, NY, USA: Association for Computing Machinery, 2022, p. 128–135. [Online]. Available: https://doi.org/10.1145/3486622.3493917

[8] S. B. Far, S. M. H. Bamakan, Q. Qu, and Q. Jiang, "A review of non-fungible tokens applications in the real-world and metaverse," *Procedia Computer Science*, vol. 214, pp. 755–762, 2022.

[9] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (nft): Overview, evaluation, opportunities and challenges," *arXiv preprint arXiv:2105.07447*, 2021.

[10] A. Gauhar, N. Ahmad, Y. Cao, S. Khan, H. Cruickshank, E. A. Qazi, and A. Ali, "xDBAuth: Blockchain based cross domain authentication and authorization framework for Internet of Things," *IEEE Access*, vol. 8, pp. 58 800–58 816, 2020, publisher: IEEE.

[11] F. Ghaffari, E. Bertin, N. Crespi, and J. Hatin, "Distributed ledger technologies for authentication and access control in networking applications: A comprehensive survey," *Computer Science Review*, vol. 50, p. 100590, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013723000576

[12] G. D. Putra, V. Dedeoglu, S. S. Kanhere, R. Jurdak, and A. Ignjatovic, "Trust-based blockchain authorization for iot," *IEEE Transactions on Network and Service Management*, 2021.

[13] A. I. Abdi, F. E. Eassa, K. Jambi, K. Almarhabi, M. Khemakhem, A. Basuhail, and M. Yamin, "Hierarchical blockchain-based multi-chaincode access control for securing iot systems," *Electronics*, vol. 11, no. 5, p. 711, 2022.

[14] F. Ghaffari, E. Bertin, N. Crespi, S. Behrad, and J. Hatin, "A novel access control method via smart contracts for internet-based service provisioning," *IEEE Access*, vol. 9, pp. 81 253–81 273, 2021.

[15] J. Zhang, Y. Yang, X. Liu, and J. Ma, "An efficient blockchain-based hierarchical data sharing for healthcare internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 7139–7150, 2022.

[16] Q. Gong, J. Zhang, Z. Wei, X. Wang, X. Zhang, X. Yan, Y. Liu, and L. Dong, "Sdacs: Blockchain-based secure and dynamic access control scheme for internet of things," *Sensors*, vol. 24, no. 7, p. 2267, 2024.

[17] L. Feng, J. Lin, F. Qiu, B. Yu, Z. Jin, J. Wang, J. Cheng, and S. Yao, "Sdac-bbpp: A secure dynamic access control scheme with blockchain-based privacy protection privacy for iiot," *IEEE Transactions on Network and Service Management*, 2024.

[18] S. Gao, G. Piao, J. Zhu, X. Ma, and J. Ma, "Trustaccess: A trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5784–5798, 2020.

[19] Y. Zhuang, C.-R. Shyu, S. Hong, P. Li, and L. Zhang, "Self-sovereign identity empowered non-fungible patient tokenization for health information exchange using blockchain technology," *Computers in biology and medicine*, vol. 157, p. 106778, 2023.

[20] K. A. Ahmed, S. F. Saraya, J. F. Wanis, and A. M. Ali-Eldin, "A blockchain self-sovereign identity for open banking secured by the customer's banking cards," *Future Internet*, vol. 15, no. 6, p. 208, 2023.

[21] N. Aryal, F. Ghaffari, E. Bertin, and N. Crespi, "A blockchain-based approach for service level agreement management in cellular network," in *15th International Conference on Network of the Future (NoF)*, 2024.

[22] J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo, and T. Chen, "Defining smart contract defects on ethereum," *IEEE Transactions on Software Engineering*, 2020.

[23] N. Aryal, F. Ghaffari, S. Rezaei, E. Bertin, and N. Crespi, "Private cellular network deployment: Comparison of openairinterface with magma core," in *2022 18th International Conference on Network and Service Management (CNSM)*. IEEE, 2022, pp. 364–366.

[24] "go-ethereum," https://geth.ethereum.org/, accessed: 2024-04-07.

[25] "Go ethereum - github," https://github.com/ethereum/go-ethereum, accessed: 2024-04-07.

[26] "Clique poa protocol rinkeby poa testnet," https://github.com/ethereum/EIPs/issues/225, accessed: 2024-04-07.

[27] S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain," p. 11.

[28] C. Dannen, *Introducing Ethereum and solidity*. Springer, 2017, vol. 1.

[29] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[30] G. Kaur and C. Gandhi, "Scalability in blockchain: Challenges and solutions," in *Handbook of Research on Blockchain Technology*. Elsevier, 2020, pp. 373–406.