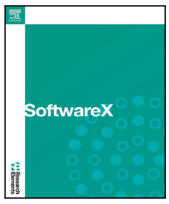
SoftwareX 28 (2024) 101932

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx



Original software publication

BiometricIdentity dApp: Decentralized biometric authentication based on fuzzy commitment and blockchain

Nibras Abo Alzahab [a,*], Giulia Rafaiani [a], Massimo Battaglioni [a], Ana Cavalli [b], Franco Chiaraluce [a], Marco Baldi [a]

[a] Università Politecnica delle Marche, Department of Information Engineering (DII), Via Brecce Bianche, 12, 60131 Ancona AN, Italy
[b] Montimage, 39 rue Bobillot, 75013, Paris, France

ARTICLE INFO

Keywords:
Biometrics
Blockchain
Decentralized authentication
Decentralized application
Fuzzy commitment scheme

ABSTRACT

As biometric authentication has been increasingly integrated into cutting-edge technology, it is interesting to study how its level of trust and interoperability across multiple devices can be increased. They can actually be enhanced through decentralization, particularly by using blockchain technology. Since transaction data on the blockchain are open and readable by all parties, a high level of user trust is achieved, enhancing transparency and interoperability across the network. The software we propose bridges the gap between the security of biometric information and the transparency of blockchain and decentralized technologies. Specifically, the software is a decentralized application (dApp), based on the Ethereum blockchain, which relies on a smart contract to manage its logic. The logic of the smart contract employs the fuzzy commitment scheme (FCS) to securely hash biometric templates, while always maintaining fault tolerance thanks to error correction codes (ECC). This mechanism ensures data integrity within a transparent, decentralized framework. The proposed dApp enhances biometric authentication by supporting both the enrollment and authentication processes. Its smart contract enables managing access control within this decentralized infrastructure. In practical applications, the proposed system can demonstrate its potential as a secure and decentralized alternative to traditional centralized systems.

Code metadata

Current code version	V 1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00379
Permanent link to Reproducible Capsule	
Legal Code License	SPDX-License-Identifier: MIT
Code versioning system used	git
Software code languages, tools, and services used	Solidity, JavaScript, Python, truffle, and Linux command line
Compilation requirements, operating environments & dependencies	Linux, Node.js, truffle, and truffle-assertions (for testing)
If available Link to developer documentation/manual	https://github.com/secomms/decentralizedbiometrics/blob/main/README.md
Support email for questions	N.ABo-Alzahab@pm.univpm.it, secomms@dii.univpm.it

1. Motivation and significance

In the digital domain, rising cybersecurity threats and vulnerabilities in centralized identity systems have driven the need for enhanced security. This urgency led to the development of the proposed Decentralized Application (dApp) [1] that includes, as its core element, a smart contract for Decentralized Identity Management.[1] Transitioning to a decentralized model, we secure identity data on a tamper-proof ledger, enhancing security and user control. Our approach enables immutable record-keeping and role-based access control, ensuring secure and context-aligned interactions. The proposed software is aimed at enabling decentralized management of biometric templates and related authentication, achieving interoperability across multiple devices

* Corresponding author.
E-mail address: n.abo_alzahab@pm.univpm.it (Nibras Abo Alzahab).

[1] A pre-print version of [1] is available at: http://arxiv.org/abs/2409.11303

https://doi.org/10.1016/j.softx.2024.101932
Received 16 July 2024; Received in revised form 20 September 2024; Accepted 8 October 2024
Available online 21 October 2024

Table 1
Comparison of different protocols for user authentication.

Proposal	Blockchain type	Storage method	Template security
[3]	Permissioned	On-chain	Fragmentation
[6]	Private	Smart cards	Encryption
[7]	Permissioned	Token	Shamir's secret sharing
[8]	Permissioned	On-chain	Fuzzy extractors
[4]	Private	On-chain	AES and MD5 hashing
[9]	Public	On-chain	Hashing using SHA256
Ours	Public or private	On-chain	Fuzzy commitment scheme

without the need to repeat initial onboarding on each individual device. It combines robust constructs for subjects and nodes, embracing Decentralized Identity Management, and improving security and user empowerment.

The topic of decentralized authentication has been extensively explored in the scientific literature; some relevant studies are summarized in Table 1. In order to clarify the novelty of our approach based on the smart contract development, a comparison between our methodology and existing approaches was conducted. There are few comprehensive studies focused on the integration of biometrics and blockchain; a notable one is [2], which discusses the costs, privacy, and security implications of blockchain for biometrics. Our approach simplifies the development of the smart contract and lowers the execution costs. Other works as [3,4] focus on off-chain storage of biometric data, while our method allows for biometric authentication using only data stored in the blockchain in a secure way.

The primary proposals related to utilizing the blockchain technology for the security of biometrics can be grouped as follows, based on the method they use to handle biometric templates [5]:

1. full on-chain storage: templates are stored onto the blockchain, using some measure to protect their confidentiality;
2. data hashing: templates are stored off-chain and their hashes are stored on-chain, providing integrity guarantees (this is also the case of [4]);
3. linked data structure: templates are used, for example, as nodes of a Merkle tree.

The novelty of the proposed scheme is adopting a hybrid approach that combines on-chain and off-chain storage to balance security, privacy, and scalability concerns. In the proposed scheme, essential cryptographic hashes and offsets are stored on-chain while the biometric data themselves remain off-chain. By only storing cryptographic information on-chain, the system ensures that the blockchain does not hold any raw or directly interpretable biometric data, protecting user privacy even in public blockchains like Ethereum. The actual biometric data (such as fingerprints, iris biometrics, electroencephalography (EEG) biometrics, etc.) are processed off-chain and deleted after a process (enrollment or authentication) is done. This ensures that sensitive biometric information is not stored or retained beyond its immediate use, which minimizes the risk of data leak, enhances user privacy, and aligns with privacy regulations such as the European Union General Data Protection Regulation (GDPR). Our approach differs radically from existing ones. In fact, in existing approaches using off-chain storage, it is necessary to retrieve the template stored off-chain each time a new authentication attempt is made, to compare it with the new acquisition. In contrast, approaches that use on-chain storage exploit hash functions or encryption to avoid compromising user privacy. Hash-based solutions are well suited to the use of mnemonic credentials such as passwords, while they do not allow a comparison of fuzzy credentials, such as biometric ones, because the digest of each acquisition of a fuzzy credential is different from the others. The use of on-chain storage along with encryption, on the other hand, makes it possible to retrieve the original template with which to make comparisons, since it is stored on-chain. However, such storage requires a lot of space, and it is also exposed to the risk of compromise of passwords or ciphers used to store encrypted credentials, resulting in their recovery. The method we propose, based on Fuzzy Commitment Scheme (FCS) [10], combines the advantages of both approaches, in that it takes advantage of on-chain storage but does not require the original template to be available (either in clear or encrypted form), as it allows it to be compared with new acquisitions directly in the digest domain, from which the template cannot be recovered.

2. Software architecture

In the complex world of digital identity management, our software, implementing the framework introduced in [1], offers an accessible decentralized way to securely store and manage biometric data on the Ethereum blockchain. The smart contract described in this paper facilitates decentralized biometric authentication based on fuzzy commitments and blockchain. This software operates in three main stages: registration, enrollment, and authentication. A block diagram of the operation environment of the smart contract is shown in Fig. 1. The theoretical background of the three stages is described in Section 2.1, followed by a technical description of the Solidity code in Section 2.2. A thorough theoretical description of the system can be found in [1].

2.1. Theoretical description

The proposed dApp implements the FCS for performing decentralized authentication, and involves different types of nodes for enrollment and for authentication stages. In particular, the system includes Enrollment Centers (ECs) and Authentication Centers (ACs). The ECs are responsible for the enrollment of new users (and, therefore, new biometrics) within the system, while ACs can perform authentication on enrolled users using their biometrics.

The mathematical operations needed for the FCS are performed off-chain to avoid submitting the clear biometric template to the blockchain and to reduce the gas required to execute the functions of the smart contract. The dApp described in this paper aims to facilitate the performance of FCS, leveraging blockchain technology for storing data and managing access control. The smart contract is deployed by an Enrollment Center, which could be, for example, a governmental body (as the scenario described in Section 3) or headquarters of an agency. The EC that deploys the smart contract is called Main Enrollment Center (MEC). The MEC has the ability to upgrade other nodes to gain powers. We note that, once the system is in operation, the MEC can also step aside, if needed, without affecting the decentralization of the system.

2.1.1. Registration stage

The registration stage is the initial phase where entities declare their intention to be part of the system. In our software, this is implicitly handled during the contract deployment. The entity deploying the contract is automatically registered as the MEC with administrative privileges. When the smart contract is invoked, the deploying address is automatically designated as the MEC, granting it ownership of the smart contract. The MEC has the authority to register the addresses of other participating centers within a mapping data structure, which stores not only their addresses, but also their names and roles, specifying whether they function as ECs or ACs. This mapping plays a crucial role in enabling the smart contract to enforce role-based access control throughout the biometric authentication process. Technically, the `nodes` mapping is responsible for maintaining the `address`, `name`, and `role` of each participating entity, with the MEC being registered as the initial node. Each node within the system is defined by an identifier (`ID`), a name, an Ethereum address (`addr`), and two boolean flags — `isAuthorised` and `isEnrollment` — that determine its role as either an AC or an EC. The `nodes` mapping is a private data structure, meaning that only the smart contract itself can modify or

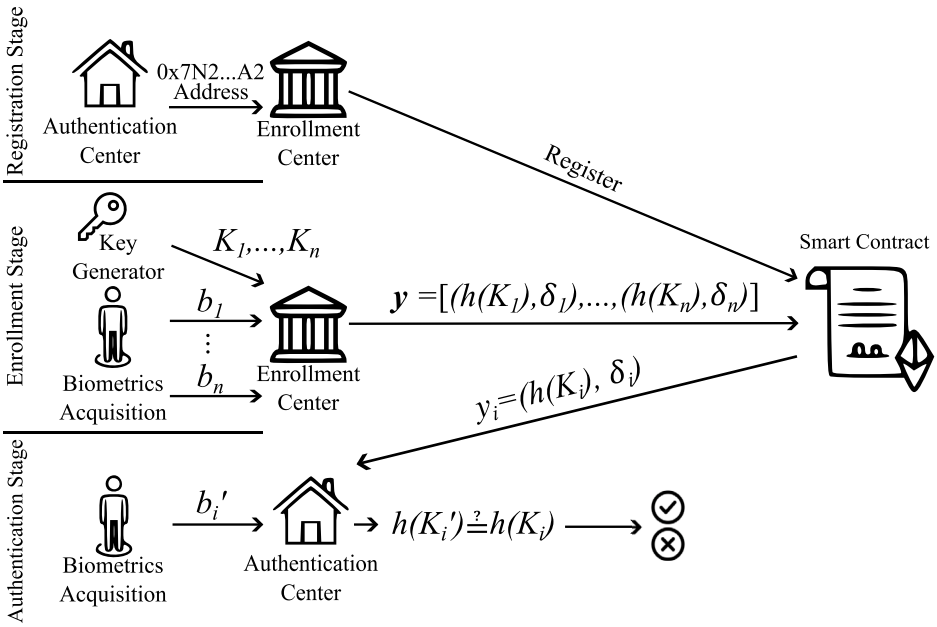


Fig. 1. Block diagram of the operation environment of the dApp [1].

retrieve nodes information, ensuring the security and integrity of the data. The MEC can add new nodes to the system using the `setNode` function, designating them as ECs. This function initializes a `Node` struct with a given identifier, name, and address. By default, any new node registered through this function is marked as authorized, allowing it to participate in the decentralized biometric authentication process.

2.1.2. Enrollment stage

In the enrollment phase, users provide their biometric data to the ECs. In our software, the ECs process the raw biometric data by extracting the features x, which represent the biometric template.

Simultaneously, a key K is randomly generated. This key is used for two purposes: firstly, it is used to generate a codeword C by using a chosen Error Correcting Code (ECC), secondly, the hash of K is generated using a hash function, as summarized next.

$$C = \mathrm{ECC}(K)$$

$$h(K) = \mathrm{Hash}(K)$$

Afterwords, the biometric template x and the codeword C are XORed to obtain the offset δ:

$$\delta = x \oplus C.$$

After that, the hash of the key $h(K)$, together with the offset δ, are used to obtain the commitment $y = (h(K), \delta)$. In this way, we ensure that the actual biometric data are not directly stored on-chain, which also enhances the privacy and the security of the subject's data. In the proposed smart contract, the `setSubject` function facilitates the enrollment, allowing ECs to submit processed biometric data.

2.1.3. Authentication stage

Once enrolled, it is needed to verify the authenticity of subjects in various scenarios. This is where ACs come into play. In fact, upon request, ACs retrieve the stored biometric data using the `getSubject` function, leveraging it to authenticate subjects based on live biometric readings. The mathematical constructs of FCS ensure that this authentication process is robust, without directly exposing the actual biometric data. Only the entities designated as ACs have the privilege to retrieve these data, ensuring controlled and secure access.

This stage works the opposite way of the Enrollment stage. The smart contract comes into play when the biometric information b' to be authenticated is read and the biometric template x' is extracted. The commitment $y = (h(K), \delta)$ is retrieved from the smart contract. The offset δ is XORed with the biometric template x' to obtain the codeword C'. The codeword is then decoded to obtain the key K', that is then hashed: $h(K')$. This hash is compared with the one retrieved from the commitment $h(K)$. If there is a match $h(K') = h(K)$, the biometric data are authenticated. To strengthen security and traceability, we implemented role-based access control in the software, differentiating between ACs and ECs using their Ethereum addresses. This ensures that only entities with the correct permissions can interact with the system in predefined ways.

In the proposed smart contract, the authentication process is facilitated through a `Subject` data structure, which includes critical components such as a unique `ID`, a hash representation (`hx`), and an offset (`delta`). These components are stored in the `subjects` mapping, which securely associates each subject's unique identifier with their corresponding data. The contract implements a set of functions that enable authorized entities to interact with the `Subject` data. ECs are empowered to create, update, and delete `Subject` entries within the `subjects` mapping, ensuring that biometric data are accurately managed and up-to-date. ACs, on the other hand, have read-only access to retrieve `Subject` data for verification purposes. These functions are strictly governed by role-based access control, enforced through modifiers that restrict execution to entities with appropriate permissions.

2.2. Software functionalities

The code defines a smart contract named `BiometricIdentity`. This contract is designed to manage biometric subject data, ECs, and ACs. The smart contract facilitates decentralized biometric authentication based on FCS and blockchain. The importance of the smart contract comes from its ability to ensure that biometric data are not only stored securely, but also remain private and accessible only to authorized entities. The smart contract is written in Solidity. It has been written and tested using the Remix Ethereum platform.[2] The key components of the smart contract are:

[2] https://remix.ethereum.org/

- **Structures**
  Two structures are defined within the contract: `Subject` and `Node`.
  - `Subject` contains information about subjects, including an ID, a hash representation (`hx`), and an offset (`delta`).

```
struct Subject {
    bytes32 ID;  // ID of the Subject
    uint16 hx;   // Hash representation
    uint16 delta;// The offset
}
```

  - `Node` represents centers and includes fields for ID, name, Ethereum address, and boolean flags indicating whether a node is authorized or serves as an EC.

```
struct Node {
    uint ID;          // Id of the Node
    string name;      // Name of the Node
    address addr;     // Ethereum address
    bool isAuthorised;// Authorization status
    bool isEnrollment;// Enrollment status
}
```

- **Private State Variable**
  The contract includes a private state variable named `owner`, which stores the Ethereum address of the contract creator.
- **Owner**
  The `owner` of the contract is the MEC, i.e., the address that deployed the smart contract. The owner can set other nodes to have its same authority and rights (ECs) and can register new ACs. We note that the owner of the smart contract can also fade away, without negatively affecting the system, thanks to its decentralized nature.
- **Mappings**
  Two mappings are declared within the contract:
  - `Subjects`: this mapping maps a string (likely subject ID) to a `Subject` structure.
  - `Nodes`: this mapping maps an Ethereum address to a `Node` structure, representing ACs and ECs.

- **Constructor**
  The constructor is executed when the contract is deployed. It sets the `owner` as the contract creator's Ethereum address and initializes a node with specific attributes, including authorization and enrollment status.

```
constructor()  {
    owner = msg.sender;
    nodes[owner] = Node({
        ID: 1,
        name: "Main_Enrollment_Center",
        addr: owner,
        isAuthorised: true,
        isEnrollment: true
    });
}
```

- **Modifiers**
  Two modifiers are defined in the contract:
  - `isAC`: This modifier checks if the caller's Ethereum address is associated with an authorized AC.

```
modifier isAC() {
    require(nodes[msg.sender].isAuthorised,
    "Caller_not_AC");
    _;
}
```

  - `isEC`: This modifier checks if the caller's Ethereum address is associated with an EC.

```
modifier isEC() {
    require(nodes[msg.sender].isEnrollment,
    "Caller_not_EC");
    _;
}
```

- **Access Control**
  Access control within the contract is based on Ethereum addresses. ACs and ECs are defined based on their respective Ethereum addresses.
- **Functions for Subjects**
  The contract provides several functions to interact with subjects:
  - `setSubjects`: allows an EC to set subject information (ID, hash, `delta`).
  - `getSubjects`: permits an authorized AC to retrieve subject information.
  - `delSubjects`: allows an EC to delete subject information.

- **Functions for Nodes** (ACs and ECs)
  Similar to subjects, functions for managing ACs and ECs are provided:
  - `setNodes`: allows an EC to set information for ACs.
  - `getNodes`: enables an EC to retrieve AC information.
  - `delNodes`: allows an EC to delete AC information.

- **Events** The contract defines a set of events to log important actions within the blockchain, ensuring transparency and traceability.
  - `SubjectSet`: logs the setting of subject information, including the subject's ID, hash representation, and offset.
  - `SubjectDeleted`: logs the deletion of subject information, including the subject's ID.
  - `NodeSet`: logs the setting of AC information, including the center ID, name, and Ethereum address.
  - `NodeDeleted`: logs the deletion of nodes (AC or EC) information.
  - `NodeStatusUpdated`: logs updates to the authorization and enrollment status of an AC, including the center Ethereum address and the updated statuses.

3. Soundness testing

In this section we illustrate the performed functional and security testing of the proposed dApp for managing biometric identities. We analyze the provided test code, highlighting the smart contract deployment, operation through its life cycle, and the robust security mechanisms it embodies.

1. **Deployment and Initialization:** The smart contract, once deployed, is rigorously tested to ensure it is correctly initialized, signifying the system readiness for operational use. The initial deployment is carried out by a MEC, ensuring control and oversight from a trusted entity in the early stage.
2. **Registration Phase:** This phase tests the contract ability to register new nodes, e.g., post offices or hospitals, into the system. These nodes are essential for the system operation, facilitating the enrollment and authentication of subjects. The successful registration emits events confirming the node addition.

3. **Enrollment Phase:** The core functionality of enrolling new biometric identities is tested here. The process involves setting subjects with unique identifiers and biometric data. Pseudo data were utilized for testing purposes, while real biometric information will be used in later versions.
4. **Authentication Phase:** This phase evaluates the system authentication mechanisms. In fact, the system should allow only the authorized nodes to access and invoke subjects' functions, while preventing any unauthorized access. The distinction between authorized and unauthorized entities highlights the contract robust access control measures.
5. **Revocation Phase:** This test ensures that only authorized entities can delete subjects' data, further reinforcing the contract security measures.
6. **Node Status Update:** Testing here focuses on the dynamic nature of node roles within the system, allowing for the update of a node status to reflect its current operational capabilities.
7. **Security Against Unauthorized Access:** A series of tests were conducted to affirm the system defence mechanisms against various unauthorized actions, including the addition, update, and deletion of subjects and nodes by unauthorized accounts. These tests validate the contract comprehensive security measures, ensuring that only authorized entities can perform sensitive operations.

The significance of these tests in enhancing smart contract security and functionality provides empirical evidence of the contract ability to securely handle sensitive biometric data, efficiently manage identities, and protect against unauthorized access.

4. Security testing

Ensuring robust security in software development is crucial, especially when dealing with sensitive information such as biometric data. Security testing is a fundamental part of the development process, as it helps identify and mitigate vulnerabilities that could be exploited by malicious actors. In the context of the BiometricIdentity dApp, rigorous security testing was conducted to ensure that the application is free from vulnerabilities and can reliably protect user data. This testing is essential for maintaining trust and ensuring the integrity of the biometric authentication system.

Three primary tools were employed for security testing: Npm audit,[3] Snyk,[4] and ESLint.[5] Each of these tools offers unique advantages and helps cover different aspects of security testing.

- Npm audit is a built-in security tool for Node.js projects that automatically checks for vulnerabilities in the dependencies used by the project. It scans the dependency tree, identifies known vulnerabilities, and provides detailed information on how to fix them. This tool is particularly useful because it integrates into the development workflow, ensuring that potential issues are caught early and promptly addressed.
- Snyk is another powerful tool specialized in identifying and fixing vulnerabilities in open source libraries and containers. It provides a comprehensive analysis of the codebase and dependencies, highlighting vulnerabilities and offering actionable remediation advice. Snyk's ability to continuously monitor for new vulnerabilities and its extensive database guarantee the maintenance of the security of the application over time.

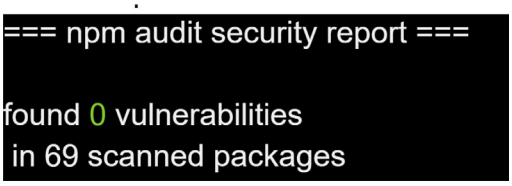


Fig. 2. Npm audit analysis results — Vulnerabilities Free.

- ESLint is a static code analysis tool used to identify issues in JavaScript code. Besides enforcing coding standards, ESLint also helps detect potential security issues, such as unsafe coding practices that could lead to vulnerabilities. By integrating ESLint into the development process, we ensure that the code adheres to best practices and is free from common security concerns.

The security testing was conducted on all pieces of code written in both JavaScript and Python, ensuring comprehensive coverage. By utilizing these tools, we performed an exhaustive vulnerability check, covering both the code and its dependencies. This multi-faceted approach ensures that the BiometricIdentity dApp is not only functional but also secure, providing users with confidence in the safety of their biometric data. As illustrated in Figs. 2 and 3, the code was found to be free from vulnerabilities, affirming the effectiveness of our security measures.

These tools collectively help in maintaining a high standard of security, contributing to the overall robustness and reliability of the BiometricIdentity dApp.

5. Interacting with the smart contract

In the realm of smart contract engagement, the initiation of transactions to activate diverse functionalities is a critical step. This interaction is expedited through the utilization of the `truffle console`[6] command within a Linux terminal environment, as depicted in 1. This method enables the execution of smart contract functions via JavaScript code snippets. Additionally, the Python programming language offers an alternative pathway for interacting with smart contracts, leveraging the Web3 library.[7] This library acts as a conduit between the Python applications and the Ethereum blockchain, facilitating seamless communication and manipulation of the smart contract.

Interactions with the smart contract can also be conducted via the Remix Ethereum platform providing a graphical user interface (GUI) that includes 15 Ethereum addresses (accounts). These accounts can be allocated to specific nodes (either enrollment or authentication nodes), facilitating the invocation of code functions under varied accounts to emulate different scenarios. This environment is useful for conducting exhaustive tests and validations of the smart contract functionalities and of its access control mechanism. The tests ensure the system efficacy in orchestrating interactions and in the robust protection of the rights and responsibilities of all the involved entities.

For the practical exploration and understanding of smart contract interactions, we have included a detailed Jupyter Notebook in the linked GitHub repository. This standalone notebook contains Python code examples for smart contract deployment and interaction. The inclusion of this notebook within our repository not only helps to clarify the process of smart contract deployment and management, but also provides a realistic hands-on interaction in using the Web3 library for interacting with the Ethereum blockchain.

This integrative approach, encompassing both the `truffle console` and the Web3 library, facilitates a multifaceted exploration and analysis of smart contract functionalities, ensuring accessibility and functioning.

[3] https://docs.npmjs.com/cli/v10/commands/npm-audit
[4] https://snyk.io/
[5] https://eslint.org/
[6] https://archive.trufflesuite.com/
[7] https://pypi.org/project/web3/

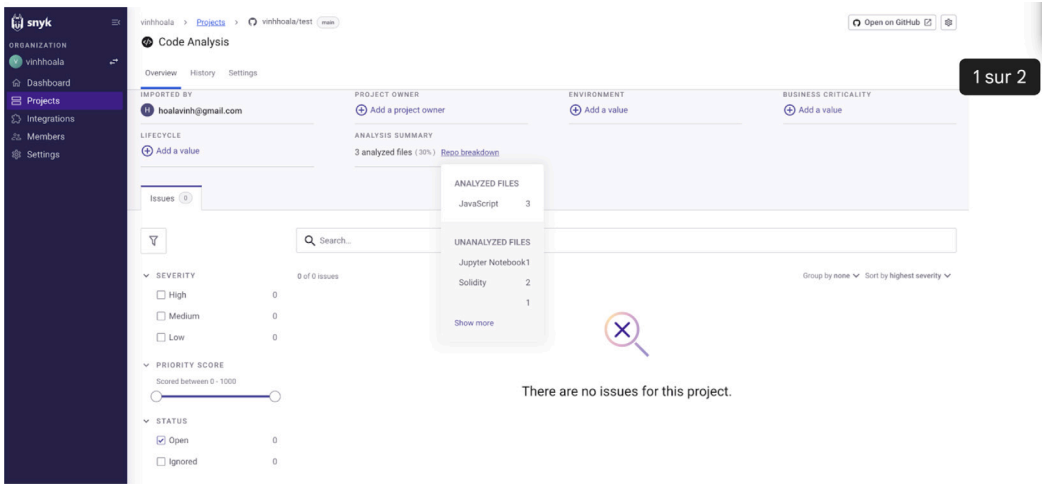


Fig. 3. Snyk code analysis results — Issues Free.

Listing 1 Interacting with the Smart Contract

```
// Truffle console output
truffle (development)> let instance;
    BiometricIdentity.deployed().then(function(inst) { instance =
    inst; });

truffle (development)> let owner = accounts[0]; let
    postOfficeAccount = accounts[1];

truffle (development)> instance.setNode(2, "PostOffice",
    postOfficeAccount, { from: owner });
tx: '0xe16a0821...3f3ee0e',
receipt: {
  transactionHash: '0xe16a0821...3f3ee0e',
  transactionIndex: 0,
  blockNumber: 10,
  blockHash: '0x1cfb523d...3949e09',
  from: '0xbf041a4...73cd4b6',
  to: '0x375801eb...78ebb08',
  cumulativeGasUsed: 22112,
  gasUsed: 22112,
  contractAddress: null,
  logs: [],
  logsBloom: '0x00000000...0000000',
  status: true,
  effectiveGasPrice: 2765281763,
  type: '0x2',
  rawLogs: []
},
logs: []
```

6. Impact

With the advent of decentralized systems and the blockchain technology, new systems can distribute control and data across a network of participants [11,12]. Unlike centralized systems, distributed systems mitigate the risk of a single point of failure and ensure that no single entity has undue control or access to the system data [13,14]. Consequently, there is the need to transfer biometric authentication into a decentralized environment, thereby reducing susceptibility to hacks and data breaches [5]. This leads to the issue of defining how sensitive data, such as biometric templates, can be effectively utilized within a transparent system like blockchain.

To address this, we have developed a dApp that is based on the FCS. The FCS ensures the secure hashing of biometric templates, while also allowing for minor tolerance through the incorporation of error-correcting codes [15,16]. FCS guarantees the secure handling of biometric data. Indeed, the piece of information that can be spread within the FCS is the commitment that, as demonstrated in [10], is hard to open for an attacker. In fact, assuming a standard value of $k = 80$, being k the dimension of the code used in FCS, and using classic assumptions for hash functions, an attacker would need 2^{79} hash computations to succeed, which is comparable to the computational effort needed to break widespread well-known cryptographic algorithms or schemes, such as RSA-1024 or SHA-1. In the proposed dApp, we store in the public blockchain only the commitment that, as we just said, is extremely hard to open for an attacker and, therefore, can be public. Using blockchain, however, increases the trust among the participants in the system and increases the availability of the data needed for biometric authentication.

The successful implementation of this system requires a robust infrastructure, which is where our smart contract plays an integral role. The primary function of the proposed smart contract is to store and manage commitments, thereby ensuring data integrity and accessibility within a decentralized framework, while ensuring transparency and security. Thanks to the immutable nature of blockchain, the proposed smart contract guarantees that once a commitment is made, it cannot be changed, thus providing a reliable record of biometric data. However, it is worth mentioning that the proposed dApp

preserves the right to be forgotten as required by the European Union GDPR. Privacy and security of the subject's data are preserved, since the actual biometric information is never stored directly on-chain. In fact, what we store on-chain is an unintelligible and irreversible reference of the biometric data that cannot be linked to the person who produced them.

Additionally, the dApp facilitates the enrollment and authentication processes, serving as an intermediary that verifies the legitimacy of the data without compromising their confidentiality. Moreover, it introduces role-based access control, distinguishing between ACs and ECs based on Ethereum addresses. This ensures that only authorized entities can interact with the system in predefined ways, thereby adding an extra layer of security and traceability.

The registration process in the smart contract is essential for preserving the security and integrity of the decentralized biometric authentication network. In the registration phase, the MEC has the exclusive power to register new nodes. Once the system is in use, ECs have the

power to add new nodes (both ACs and ECs) to the system; a voting mechanism can be added to the system to decide whether a node should be added or removed from the system. This guarantees that only entities validated by the ECs are allowed to participate in the network, thus preventing unauthorized or possibly malicious actors from obtaining access. The ECs enforce rigorous control measures, exclusively registering nodes that have completed comprehensive verification procedures. Through the implementation of this selective registration, the system guarantees that only reliable nodes are allowed to participate, hence upholding the overall security and dependability of the decentralized authentication process. The importance of this technique is particularly important in a decentralized setting, since the reliability of the entities involved directly affects the system ability to withstand security risks.

In terms of ownership, deploying the smart contract should be performed by an organizational entity rather than individuals. The invoker of the smart contract can assign the roles of ECs and ACs to different organizations or entities. Thanks to the function `deleteNode`, any node can be forgotten from the dApp list of nodes. Even the owner can be deleted, leaving the rest of the ECs to operate in a decentralized manner. To elucidate the application of this software, consider the following scenarios:

- In the first scenario, a governmental entity (e.g., a ministry) deploys the smart contract, thereby becoming the owner and an EC. The owner node can then register various organizations, such as post offices, hospitals, and municipalities, as ECs or ACs through the functions `setNode` and `updateNodeStatus`. These organizations would then interact directly with people; in fact, the ECs can proceed to enroll new subjects in the system simply by collecting people biometric data using the function `setSubject`. Subsequently, the ACs can retrieve the enrollment commitments for authentication purposes through the function `getSubject`, every time a subject wants to be authenticated.

- In the second scenario, a company headquarters deploy the smart contract and become the owner, with their branches serving as ECs and ACs. In this case, the ECs would enroll the employees, acquiring their biometrics.

These examples emphasize the great scalability of the proposed dApp, ranging from a few nodes to a national or even international scale. Moreover, the more nodes there are, the more secure the system becomes, as for any blockchain-based application.

Another point that is worth discussing is the scalability of our solution. When adding more nodes to the network, it is crucial to take actions that require the system to be highly scalable. The main goal is to reduce the workload directly carried within the blockchain system by offloading computationally demanding operations to external processes, which helps to relieve its load. This approach guarantees security and maintains data integrity while simultaneously efficiently lowering costs and improving transaction processing speed by giving first priority to the blockchain basic operations. We increase speed and cost-effectiveness by contracting out complex computations. Furthermore, the system architecture uses blockchain data storage, keeping only essential elements such as offsets and cryptographic hashes. This improves data management efficiency and helps to preserve memory. This approach improves the system potential to manage large-scale deployments, where efficiently managing the significant volume of data and transactions would normally be challenging. However, constant improvement is necessary to maximize the advantages of this architecture on a large scale. While scaling to suit the increasing user population, our main goals should be to maximize the efficiency of on-chain data and simplify off-chain processes to guarantee system performance and cost-effectiveness.

7. Conclusion

In this era of rapidly advancing technology, the integration of security and transparency is essential for establishing and maintaining trust among individuals. Decentralization plays a key role in this dynamic, as it empowers users by distributing control and eliminating the reliance on centralized entities, making systems more secure, transparent, and interoperable. In this paper, we propose a comprehensive solution for decentralized biometric authentication in the form of a decentralized application (dApp), which integrates multiple cutting-edge technologies: blockchain for secure, decentralized and transparent data management, and the fuzzy commitment scheme (FCS) for privacy-preserving biometric authentication.

The dApp we propose encompasses various critical stages, including registration, enrollment, authentication and revocation, providing a complete lifecycle approach to managing digital identities based on interoperable biometric credentials. During the registration and enrollment stages, biometric data are acquired and processed through the FCS, ensuring that sensitive information remains safe. Blockchain technology, with its immutable ledger and decentralized nature, ensures that data are not only tamper-proof but also traceable, allowing users to retain control over their personal information. A unique feature of the solution we propose is the way it handles data security and privacy. Biometric data are encrypted and securely stored, with access granted only to authorized parties. Data are never exposed to potential vulnerabilities associated with public storage, since FCS ensures a high level of protection. This approach guarantees that users' biometric information remains private and accessible only to those with proper permissions.

The application of these technologies goes beyond traditional security methods, as they offer a foundation for future-proof digital identities that are both robust and reliable. By leveraging blockchain reliability along with FCS cryptographic protection, the proposed solution addresses the key challenges in managing digital identities: security, privacy, and accessibility.

Ultimately, this dApp provides a solution to the ongoing concerns surrounding digital identity management in the modern world. Individuals can rely on their digital identities without fear of unauthorized access or data breaches, while they can benefit from a distributed and interoperable system. The combination of these technologies ensures that digital identities are not only secure but also resilient, making them adaptable to the evolving digital landscape.

CRediT authorship contribution statement

Nibras Abo Alzahab: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Giulia Rafaiani:** Writing – review & editing, Writing – original draft, Visualization, Conceptualization. **Massimo Battaglioni:** Writing – review & editing, Writing – original draft, Validation, Resources, Data curation, Conceptualization. **Ana Cavalli:** Writing – original draft, Validation, Software, Resources. **Franco Chiaraluce:** Validation, Supervision, Resources, Funding acquisition, Conceptualization. **Marco Baldi:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Investigation, Funding acquisition, Conceptualization.

Funding

This work was partially supported by project TRUST "digital TuRn in EUrope: Strengthening relational reliance through Technology" under the Horizon 2020 – MSCA (Marie Skłodowska-Curie Actions) – RISE (Research and Innovation Staff Exchange) program [101007820, 2021], and by project SERICS, Italy (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, and by FSC project "Pesaro CTE SQUARE", Italy, funded by MIMIT, CUP D74J22000930008.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Nibras Abo Alzahab reports statistical analysis, travel, and writing assistance were provided by Montimage. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was conducted in collaboration between Università Politecnica delle Marche (UNIVPM) and Montimage. The experimental work was mainly conducted at Montimage.

Data availability

No data was used for the research described in the article.

References

[1] Abo Alzahab N, Rafaiani G, Battaglioni M, Chiaraluce F, Baldi M. A decentralized biometric authentication based on fuzzy commitments and blockchain. In: Proceedings of the 6th international conference on blockchain computing and applications. 2024.

[2] Delgado-Mohatar O, Fierrez J, Tolosana R, Vera-Rodriguez R. Blockchain and biometrics: A first look into opportunities and challenges. In: International congress on blockchain and applications. Springer; 2019, p. 169–77.

[3] Lee YK, Jeong J. Securing biometric authentication system using blockchain. ICT Exp 2021;7(3):322–6.

[4] Mohsin AH, Zaidan A, Zaidan B, Albahri OS, Albahri AS, Alsalem M, et al. Based blockchain-PSO-aes techniques in finger vein biometrics: A novel verification secure framework for patient authentication. Comput Stand Interfaces 2019;66:103343.

[5] Delgado-Mohatar O, Fierrez J, Tolosana R, Vera-Rodriguez R. Blockchain meets biometrics: Concepts, application to template protection, and trends. 2020, arXiv preprint arXiv:2003.09262.

[6] Páez R, Pérez M, Ramírez G, Montes J, Bouvarel L. An architecture for biometric electronic identification document system based on blockchain. Future Internet 2020;12(1). http://dx.doi.org/10.3390/fi12010010, URL https://www.mdpi.com/1999-5903/12/1/10.

[7] Nandakumar K, Ratha N, Pankanti S, Darnell S. Secure one-time biometrie tokens for non-repudiable multi-party transactions. In: IEEE workshop on information forensics and security. 2017, p. 1–6. http://dx.doi.org/10.1109/WIFS.2017.8267654.

[8] Bao D, You L. Two-factor identity authentication scheme based on blockchain and fuzzy extractor. Soft Comput 2021;1–13.

[9] Kaur V, Bhatt DP, Tharewal S, Tiwari PK. Blockchain-based secure storage model for multimodal biometrics using 3D face and ear. In: 2023 international conference on advancement in computation & computer technologies. IEEE; 2023, p. 860–5.

[10] Juels A, Wattenberg M. A fuzzy commitment scheme. In: Proceedings of the 6th ACM conference on computer and communications security. 1999, p. 28–36.

[11] Bao Q, Li B, Hu T, Sun X. A survey of blockchain consensus safety and security: State-of-the-art, challenges, and future work. J Syst Softw 2023;196:111555.

[12] Wenhua Z, Qamar F, Abdali T-AN, Hassan R, Jafri STA, Nguyen QN. Blockchain technology: security issues, healthcare applications, challenges and future trends. Electronics 2023;12(3):546.

[13] Massoud MA, Mokbel M, Alawieh S, Yassin N. Towards improved governance for sustainable solid waste management in Lebanon: Centralised vs decentralised approaches. Waste Manag Res 2019;37(7):686–97.

[14] Sreeramareddy CT, Sathyanarayana T. Decentralised versus centralised governance of health services. Cochrane Database Syst Rev 2019;2019(9).

[15] Rathgeb C, Uhl A. Adaptive fuzzy commitment scheme based on iris-code error analysis. In: 2010 2nd European workshop on visual information processing. IEEE; 2010, p. 41–4.

[16] Chauhan S, Sharma A. Improved fuzzy commitment scheme. Int J Inf Technol 2022;14(3):1321–31.