symmetry

MDPI

Article

Securing Fingerprint Template Using Blockchain and Distributed Storage System

Moses Arhinful Acquah [1], Na Chen [1], Jeng-Shyang Pan [2], Hong-Mei Yang [2] and Bin Yan [1,*]

1 College of Electronics and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China; Arhinful8@yahoo.com (M.A.A.); nchen@sdust.edu.cn (N.C.)
2 College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; jspan@cc.kuas.edu.tw (J.-S.P.); skd991737@sdust.edu.cn (H.-M.Y.)
* Correspondence: yanbinhit@sdust.edu.cn

Received: 25 May 2020; Accepted: 3 June 2020; Published: 4 June 2020

Abstract: Biometrics, with its uniqueness to every individual, has been adapted as a security authentication feature by many institutions. These biometric data are processed into templates that are saved on databases, and a central authority centralizes and controls these databases. This form of storing biometric data, or in our case fingerprint template, is asymmetric and prone to three main security attacks, such as fake template input, template modification or deletion, and channel interception by a malicious attacker. In this paper, we secure an encrypted fingerprint template by a symmetric peer-to-peer network and symmetric encryption. The fingerprint is encrypted by the symmetric key algorithm: Advanced Encryption Standard (AES) algorithm and then is uploaded to a symmetrically distributed storage system, the InterPlanetary File system (IPFS). The hash of the templated is stored in a decentralized blockchain. The slow transaction speed of the blockchain has limited its use in real-life applications, such as large file storage, hence, the merge with IPFS to store just the hashes of large files. The encrypted template is uploaded to the IPFS, and its returned digest is stored on the Ethereum network. The implementation of IPFS prevents storing the raw state of the fingerprint template on the Ethereum network in order to reduce cost and also prevent identity theft. This procedure is an improvement of previous systems. By adopting the method of template hashing, the proposed system is cost-effective and efficient. The experimental results depict that the proposed system secures the fingerprint template by encryption, hashing, and decentralization.

Keywords: blockchain; IPFS; fingerprint; hashing; decentralization; biometrics; encryption

1. Introduction

Blockchain is amongst the emerging technologies that have a relatively strong cryptographic foundation that enables applications to leverage its features to achieve resilient security solutions [1–3]. The fingerprint, which is a form of biometrics, is used for security authentication in most high-level security institutions. The traditional ways of protecting one's privacy, such as passwords, tokens, and key code, have been slowly eliminated with the introduction of how unique and secure the fingerprint is with every individual. The fingerprint template is usually stored in a centralized database, which raises the risk of spoofing, data tampering, identity theft, and channel interception [4]. Researchers have previously proposed various systems to solve this problem in order to make the system more secure, efficient, and decentralized, and they implemented the blockchain technology. This technology, introduced as the technology behind Bitcoin cryptocurrency, has features, such as immutability and decentralization, and incorporated into the new system. The identity document (ID) is an essential document of every citizen. This property is still vulnerable to security flaws, as mentioned earlier. Hence, it is the introduction of a biometric electronic identity document (e-ID) [5]. The embedding of fingerprint or signature of the citizen and later secured by the blockchain technology's transaction validation makes it more immutable [5–7]. The user identity management system consists of three parts: a user, a registration center, and an authentication server. It utilizes a permissioned blockchain system (Permissioned blockchains requires access to join, taking the features of the blockchain but not giving up the central authority figure) for key management to make it tamperproof [8]. The user identity information is then uploaded onto the system in its raw state. However, using a raw state might increase the blockchain's size over a certain period, leading to the issue of the bottleneck of the blockchain. Using different fingerprint recognition techniques results in different vector templates sizes [9–12], but none is compared to the hash of all the templates derived from the recognition techniques. This is because IPFS uses SHA256, which results in a 32 bytes hash. As a result of the bottleneck, this system requires a high cost for storing the data and high computational power [1].

When considering the security vulnerabilities that are associated with centralized data storage, the use of distributed and decentralized technology to eliminate central authority is essential and risk-free [13]. Using the Ethereum network in our proposed system, data are stored on nodes that keep a copy of all transactions dating back to the first one in a chain of blocks. The hash of the previous block connects each block; hence, any tampering will be noticed. This system ensures data security when a node or multiple nodes are under attack. The merging of the fingerprint template or biometrics with the Inter-Planetary file system (IPFS) and Ethereum solves the high cost of storing raw data on the blockchain [1]. The security risk, such as spoofing by a malicious node, is still adamant [4]; this is because, when data are stored on the IPFS, they send chunks to every node, which can still be accessed by a malicious node [14]. Therefore, encrypting the fingerprint template after pre-processing is essential before uploading it to IPFS. Data hashing technique by the IPFS is somewhat unsecured, because data can be retrieved if in possession of the hash [15]. Accordingly, by implementing the symmetric key; advanced encryption standard (AES) or any other encryption algorithm to files before uploading to the IPFS solves the issue of system viability when compromised by a malicious node or attacker is rectified. We concluded that incorporating the technology would be beneficial to our proposed system due to the significant difference of blockchain as a database and the traditional fingerprint database, as shown in Figure 1.

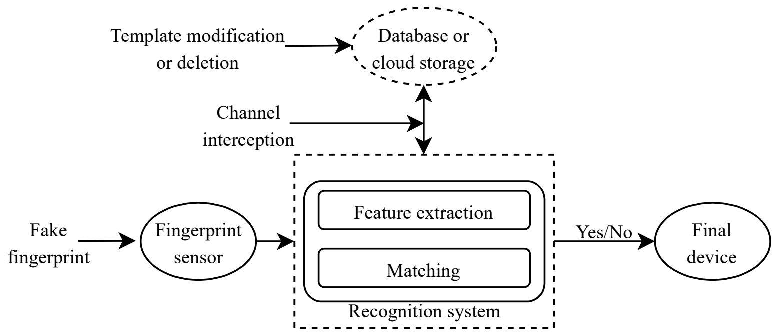

Figure 1. A Traditional fingerprint authentication system.

The proposed system by this paper explores the use of peer-to-peer symmetric system to secure fingerprint templates, and contributes in the following ways:

1. A merge of Ethereum and IPFS architecture proposal for decentralized fingerprint storage.
2. By using data hashing, the system becomes cost-effective and efficient.
3. Vulnerabilities from distributing templates on IPFS is tackled by encryption.

This paper hereafter is organized, as follows: Section 2 presents the background of this paper. Section 3 explains related works on blockchain and IPFS with fingerprints, other biometric features, and content security. In Section 4, we describe the proposed architecture and algorithm vividly.

Section 5 focuses on the experimental details and results. In Section 6, the paper ends by concluding and some challenges.

2. Background

Fingerprint templates are extracted and needed for authentication in numerous security required institutions due to their uniqueness with every individual. These templates are stored in a centralized framework and managed by a central authority, as shown in Figure 1. This system increases the risk of templates modification and channel interception by an outside attacker or by DBA. The blockchain technology has gained popularity due to the advantages that are mentioned in Table 1. On its applications in IoT, medicine, and economics outside its prior use, the Bitcoin [1]. These applications are limited because of how expensive it is to scale the system with blockchain. The blockchain ensures data integrity and anonymity. That is why its use extends to other applications that require the elimination of the central authority [1,3,16]. We chose to implement the feature of data hashing from IPFS and the Ethereum blockchain network for our experimental purpose.

Table 1. The difference between Blockchain and Traditional database.

	Blockchain	**Traditional Database**
Architecture	Every node on the network has a copy of transactions, so with every change that occurs, there will be discrepancies.	Records secured on databases are centralized and risk of attacks.
Authority	Using smart contracts and consensus algorithms, nodes on the network are made to trust each other.	The use of a database administrator (DBA) eliminates trusts with participating parties because the DBA controls data and can modify or delete data.
Security	No single point of failure.	The database is prone to malicious attacks since its centralized.

2.1. Blockchain Technology

Blockchain technology has grown successfully since the launching of Bitcoin as a digital currency. Satoshi Nakamoto introduced the Bitcoin mechanism in 2008 in a paper Bitcoin: A Peer-To-Peer Electronic Cash System [2]. The Bitcoin is simply a peer-to-peer version of the electronic cash that enables instant online transactions from one user to another without a financial institution. The technological idea behind blockchain is related to the database, except communications with them differ. A blockchain is a distributed database of records of all transactions that have been completed and distributed amongst participating nodes while using a consensus algorithm [17]. The nodes on the blockchain add blocks containing transactions linked together using the hashes of previous blocks that are shown in Figure 2.

The Ethereum blockchain was implemented in our proposed work as a type of public blockchain because it is an open-source platform that allows decentralized applications (DApps) to run on it, unlike the Bitcoin [18,19]. The Ethereum network uses ether (ETH) as its cryptocurrency with unique addresses of the prefix “0x”. This platform utilizes smart contracts, which are self-executing codes that are deployed to the network. The simpler the smart contract, the less costly it is to deploy and execute functions to interact with DApps. The Ethereum platform was proposed in 2015, and it currently has a size of 300 GB and transaction speed of 25/tps [19]. This platform uses the Casper the friendly finality gadget (FFG) [20], a hybrid of other consensus algorithms that prevent the denial of attack by nodes hosted on the network. The Ethereum Virtual machine (EVM) makes it easier to implement DApp architecture on the network.

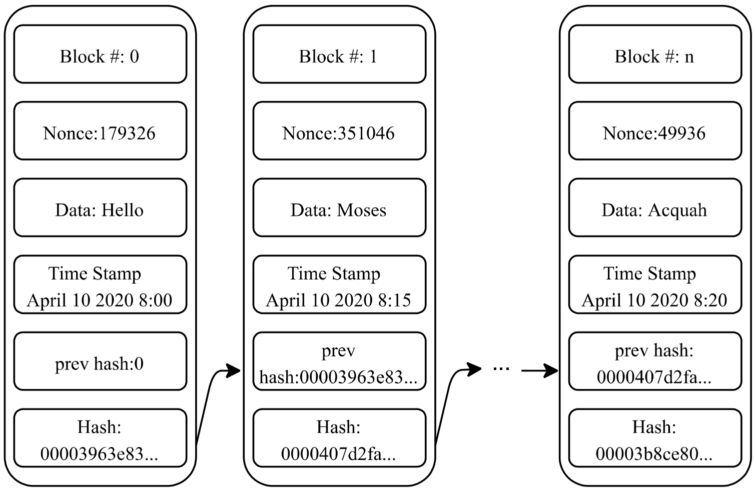

Figure 2. Typical Blocks of a Blockchain.

Where $S[0]$ is the state at the end of the previous block, and T_X is the transactions list with n transactions of the block in Figure 3.

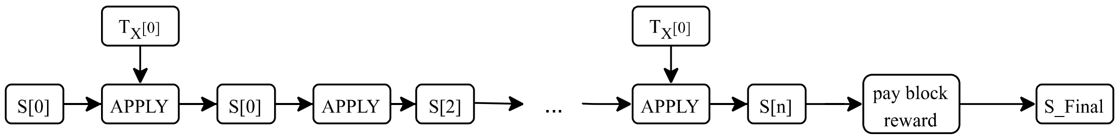

Figure 3. Block validation of the Ethereum network [18].

2.2. The Inter-Planetary File System (IPFS)

Juan Benet and the Protocol Labs created the IPFS as a quick system to move around scientific data. The IPFS has since become an open-source, permanent web protocol for storing and sharing data on a distributed file system. Content addressing (i.e., describing and obtaining content using the information of the file) is the method by which the IPFS locates the stored data [14,21]. It aims to make the web quicker, more reliable, and more accessible. IPFS could become a significant modern subsystem of the internet if upgraded; it could complement or succeed HTTP.

The Ethereum network and IPFS both use a form Merkle tree for their data structure. The Ethereum network uses the Merkle Patricia tree, which simply works by encoding the key of the value to the path that is used to take down a tree. The IPFS uses the Merkle DAG, which uses the DAG (Direct Acyclic Graph) mapping and indicates each node as an identifier. This Merkle DAG hashes the contents of each leaf node.

3. Related Works

In this section, we review the applications that are related to blockchain and IPFS in biometrics and content protection, since these are sensitive and vital information of every individual. We discovered that most existing solutions reviewed are abstract and with no or little implementation details. Accordingly, the system proposed is built on these deficiencies.

3.1. Blockchain in Biometrics

In 2018, the authors in [22] proposed a system to secure fingernail data where the management center is simulated while using node.js. This system comprises a data management system that acts as nodes responsible for storing fingernail data sent by device nodes, which is responsible for pre-processing the captured fingernail from the user. The idea of off-chain (i.e., processing data off the blockchain) is adapted into this paper because of the workload that is incurred by nodes if they were storing and processing the fingernail template. Shih et al. proposed system use the "full chain" technique (i.e., pre-processed data is stored in all the data centers) of storing the fingernail template.

The concept of using the "side chain" of processing data is discussed in [13], which combines the users' fingerprint template and other user data and saves it on the side chain, which is then compared to the "Aadhar card number" on the "main chain." This method is effective in eliminating congestion. The authentication of a user is done by comparing the details on the Ethereum network.

Patients are authenticated by using the blockchain network to store encrypted hybrid patterns of the patient [23]. This hybrid pattern consists of the RFID and the finger-vein feature of the patient processed and hashed while using the MD5 and AES algorithm. This makes attacks, such as brute force attack and spoofing, near to impossible. The processed data are sent from the access node to the blockchain to be stored and later authenticated if a patient enrolls.

The use of a biometric e-ID in a system to validate users during voting using the blockchain technology was proposed in [5] to solve the malicious attack on the previous system. This system is secure, because citizen information is combined with fingerprint data and later uploaded onto the blockchain network, in order to authenticate registered citizens.

3.2. Blockchain in Content Protection

BlockIPFS was introduced by Nyaletey el al. [24] to secure the authorship of data uploaded to IPFS. The system integrates the Ethereum network and IPFS, due to the efficiency, traceability, and security it possesses [14]. The service by the BlockIPFS ensures that files uploaded to the distributed file system can be audited, traced, and improve data trustworthiness.

In [25], the authors deal with data tampering of the cloud data storage by merging the IPFS and Ethereum network. The system eliminates the need for trusting the cloud storage provider (CSSP), which acts as an authority figure with the power to manipulate or sell data for its benefits.

A decentralized architecture and solution were proposed by authors in [21] in order to control the version of documents being shared amongst users. This system secures the basic features of blockchain, IPFS, and smart contracts to make it wholly decentralized and tamperproof, preceding the need for a cloud or centralized database. The smart contract is responsible for interactions between participants and it handles the registration of users and approvals of new versions on the IPFS.

The above-proposed systems are unique in their way; they commonly eliminate the centralization of data storage. With the implementation of IPFS, some systems are cost-effective in storing the raw data on the Ethereum network. The essential feature some of these systems lack is data encryption before uploading it on the IPFS. Even with encrypting data in [23], the system stored the data on the blockchain network; this is also costly. That is why our proposed system encrypts the fingerprint template and using IPFS stores the hash on the Ethereum network. The proposed system performs similarly to the traditional fingerprint authentication system proposed by Dipti et al. [13]. This system authenticates users by matching their "Aadhar card" number against fingerprint features stored on the blockchain. In summary, Dipti's proposed system is different from ours in these ways:

1. Dipti's side chain is centralized, and our system is distributed.
2. Dipti's fingerprint and Aadhar number are not encrypted, while the proposed system secures data using the AES algorithm.
3. The cost that is involved in implementing the system is high in Dipti's because the raw fingerprint vector is stored on the blockchain. In contrast, our system is cheaper by using only the IPFS hash.
4. Our system is scalable and efficient because of the size of the hash, while Dipti's system stores extensive data in the form of a fingerprint vector on the blockchain.

4. Proposed Architecture

To resolve the issues that are listed in the previous section, our proposed system $\mathcal{R}$, integrated with IPFS and Ethereum network, which is immutable, accessible, available, and auditable [2,14].

We encrypted the processed fingerprint template **E**, using the AES algorithm to secure the template before uploading it unto the IPFS, in order to implement the immutable feature of the whole system. This stage makes the template useless if in any chance acquired by a malicious attacker because it is near impossible to decrypt the template without having the decryption key even by brute force attack.

In this system, preprocessing and encryption of the fingerprint template is done "off-chain" in order to reduce the bottleneck of the system. The IPFS works as a distributed file system, which reduces the gas price involved when deploying smart contracts and executing commands on the Ethereum network illustrated in Figure 4. The IPFS, after successfully storing the encrypted fingerprint template, returns a unique 46-character hash. Subsequently, $\langle_g$, with the use of a smart contract, is uploaded unto the Ethereum network. For authentication, the user data $\langle_g$, returned from the Ethereum network, is compared with $\mathcal{O}$. This system flow is displayed in and Algorithm 1 and Algorithm 2.

Algorithm 1 Generate user data $\langle_g$ from **E**.

Input:
The AES encrypted template **E**.
Output:
The returned user data $\langle_g$.

```
1.  for uploading E to Ethereum network do
2.      Convert E to buffer.
3.      Upload buffer to IPFS.
4.      The IPFS returns $\langle_g$.
5.      Write $\langle_g$ to smart contract assigned $a$
6.      Check for the status of the transaction.
7.      if transaction successful = true then
8.          return $\langle_g$.
9.      else.
10.         show error.
11.     end
12. end
```

Algorithm 2 authenticate user data $\langle_g$

Input:
Generated user data $\langle_g$.
Hash values in the block chain: $\mathcal{O} = \{\langle_o^1, \langle_o^2, \langle_o^3, \ldots, \langle_o^n\}$
Output:
Verification decision V.

```
1.  for $i \in \{1,2,\ldots,n\}$
2.      if $\langle_g = \langle_i$
3.          set $V = 0$
4.          show $\langle_g$ already in the system
5.          return
6.      end
7.  end
8.  set $V = 1$
9.  add new $\langle_g$ to the system.
```

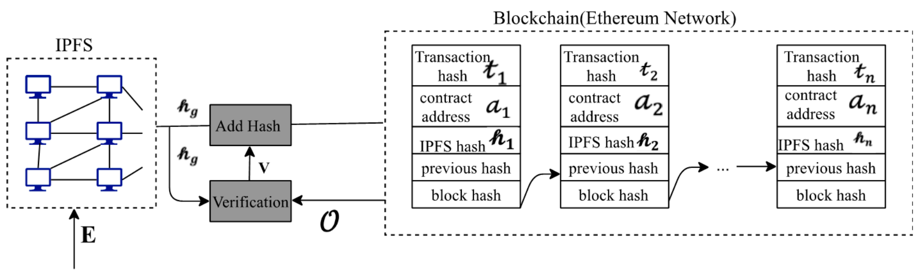

Figure 4. Proposed system architecture $\mathcal{R}$.

4.1. Authenticate User Data $\langle_g$

According to Algorithm 2, the user data $\langle_g$ is compared with all user data in $\mathcal{O}$ for verification in our proposed system. Consider this, $\mathcal{O} = \{\langle_o^1, \langle_o^2, \langle_o^3, \ldots, \langle_o^n\}$ is the original user data already saved in $\mathcal{R}$. while using Equation (1). Verification is done in the system to indicate whether $\langle_g$ is present in the system and added if not. This verification procedure can also be used to verify whether new data are manipulated or fake.

$$V = \begin{cases} 1, \, if \, \langle_g \notin \mathcal{O} \\ 0, \, if \, \langle_g \in \mathcal{O} \end{cases} \tag{1}$$

4.2. Fingerprint Feature Extraction Methods

In this section, we discussed other forms of fingerprint recognition and how templates are acquired. The fingerprint is a unique biometric feature to every individual and, hence, its adaptation into most security authentication systems. Therefore, many researchers have developed methods to acquire different features from recognition systems for matching. Templates from the different methods of recognition have different sizes of vectors, which determines the template size. In the subsequent sections, we show the cost, efficiency, and performance involved when uploading other templates that are acquired from different fingerprint features compared to our prosed system. The typical methods of fingerprint feature extraction can be classified into two:

- Minutiae-based matching: this is the most popular technique used by most researchers. This method uses minutia extracted from two fingerprints and are compared. Different approaches on how the minutia are extracted and compared. Some by using bifurcation method (BM) [11], enhancing fingerprint image and using crossing number method (MA) [10], and recognition using both local and global structures (MP) [9].
- Texture classification: basic fingerprint patterns are matched with other stored templates. This mostly uses the central point of the fingerprint and aligned with different templates. They use the discrete wavelet method (DWT) in [12] to attain a smaller vector size template. We also acquired the vector template from using the Fingercode approach by the authors in [26]. This vector template has a template vector size of 640 by using eight Gabor filters and 80 sectors. In a recent study conducted by authors in [27], we discovered the vector size of the algorithm to be large because it proposes using Fingercode, phase-only correlation, VeriFinger, and NBIS.

We decided to implement methods BM, MA, MP, and DWT because of the size of the template vector and how less time consuming it is to upload the vector template to the Ethereum network. The vector template that was acquired from the Fingercode approach and the approach proposed by authors in [27] is larger than the vector template of MA.

4.3. Design and Implementation

This section provides a detailed simulation broken down into algorithm dependencies results and discussions. For simulation purposes, we used the EVM to execute smart contracts according to Appendix A. These terms are listed below:

- GasPrice: the amount of wei the sender is willing to pay per unit of the gas required to execute the transaction.
- GasLimit: The maximum amount of gas the sender is willing to pay for executing this transaction is set before any computation is done.
- Data: the input data (fingerprint template) of the message call.
- sendHash: the owner executes this function. An event is triggered to upload the Hash $\langle_g$ of the template, and it is stored on the smart contract.

- getHash (The cost only applies when called by the storeHash smart contract): returns the template hash stored on the Ethereum contract. This hash $\langle_g$ begins with "QmY ... " depending on the template, so it is identifiable whenever returned.
- deleteHash: this function is executed to delete the stored hash $\langle_g$ on the Ethereum network.

5. Results and Discussion

We discuss the experimental results in this section. First, the parameters set up to simulate our proposed system. Next, the performance metric of $\mathcal{R}$. Third, the security analysis of $\mathcal{R}$ showing improvement in the traditional fingerprint system. Finally, a comparison of our proposed system $\mathcal{R}$ to Dipti's proposed system.

5.1. Parameter Setup

In this section, we present the necessary favorable run-time environment in order to effectively simulate our proposed system, as shown in Table 2. These parameters are further explained below:

- Node.js: Node.js is a JavaScript run-time setting which incorporates everything needed to program in JavaScript
- Ganache: Ganache is a closed personal blockchain with ten accounts loaded with 100ETH each that allows smart contracts to communicate.
- Metamask: Metamask is a browser extension that makes it easier for DApps to run in your browser and interact with the Ethereum network.
- The remix compiler: this is an open-source tool for writing, compiling, debugging, and deploying smart contracts connected to metamask using web3.js.
- Solidity: a programming language for smart contracts.
- Bootstrap and JavaScript: for creating a front-end GUI.
- Local IPFS: setup a local IPFS node on using ipfs daemon on the run-time environment.

Table 2. Implementation blueprints.

Environment	Parameters
System	Intel Core i5 CPU @2.7 GHz 8 GBRAM 128 HDD OSX Catalina
Blockchain	Ganache Metamask Remix compiler Solidity 0.6.1 Node.js 13.5.0 Web3.js Bootstrap 3.3.7
IPFS	Local IPFS

5.2. Performance Metric

In this section, we discuss performance in terms of efficiency, and execution time that is involved in sending the hash of the template against different methods of template acquisition. We also discuss the cost involved in executing the functions of the storeHash smart contract.

The designed system used the ganache platform as a local blockchain set at 20 gwei to optimize the transactions. The following data are based upon gas price of 20 gwei (1 gwei = 10^{-9}ETH). The conversion rate of gwei is 1ETH = \$199.91 (May 2020), according to the ethgasstation. The ETH spent according to the gas price is higher than 2 gwei, which is the standard optimization of the remix IDE. We compared the template acquired from different fingerprint recognition techniques to illustrate efficiency, cost, and execution time against our proposed system $\mathcal{R}$. Using Equation (2), we calculated the efficiency improved from just sending the hashes of the different templates when compared to storing the raw template on the ethereum network. These efficiencies are calculated, and the results are shown in Table 3. The efficiency of using the template MA to $\mathcal{R}$ is enormous than using DWT to $\mathcal{R}$. These results show that, when using the 32 bytes hash generated by IPFS, the cost that is involved in our proposed system is lower than the varying size of the fingerprint template vector. The smart contract deployment is a one-time process, so, the system performance will not be affected, but executing the sendTemplate/Hash function varies.

$$\eta = \frac{\gamma - \beta}{\beta} \times 100 \tag{2}$$

Table 3. Efficiency of proposed $\mathcal{R}$ with storing template from different fingerprint techniques.

Technique	**Efficiency**
DWT	182.47%
MP	231.83%
BM	672.89%
MA	3927%

Where η is efficiency, β is gas used when sending $\langle_g$ with $\mathcal{R}$, and γ is the gas used sending template of other recognition techniques. Figure 5 illustrates the performance in terms of efficiency of the proposed system $\mathcal{R}$.

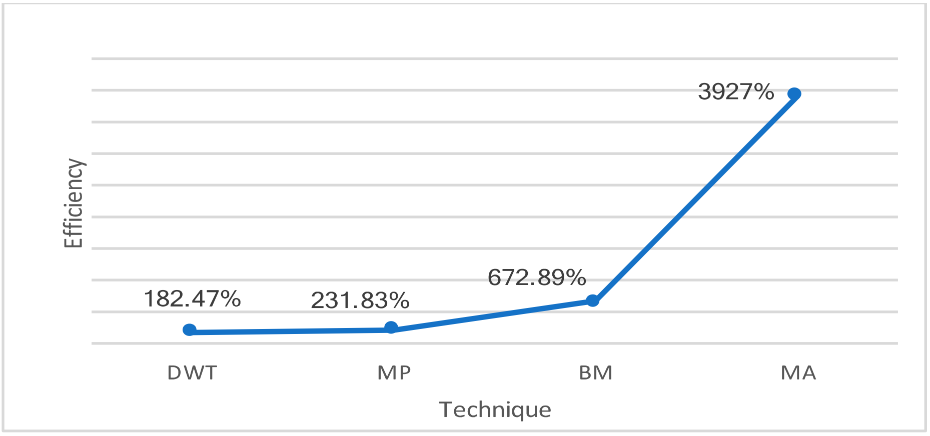

Figure 5. Efficiency of other fingerprint techniques against $\mathcal{R}$.

This fingerprint system using the storeHash contract is deployed on the Ethereum network. For deployment of any contract, a specific gas limit is set; by default, it is 3,000,000. The gas consumed for execution and transaction according to Table 4 displays the highest gas consumed is by the sendTemplate/Hash function, as uploading the template or hash needs gas. The amount of gas varies depending on the size of the smart contract. The getTemplate/Hash and deleteTemplate/Hash show the least cost because they do not require additional upload, but rather retrieval and deletion of $\langle_g$ during authentication and suspect of malicious activity. The execution costs are dependent on the lines of codes and logical operation being executed by the storeHash contract. The storeHash contract was deployed to a at 0xa8b3AfF482510a18A0e4c88F15DE214CB1397014 and using the transaction log from the ganache we can verify a successful transaction. It is a simple contract, which has not been optimized and does not tackle the user's full registration, and it is solely for experimental purposes.

Table 4. The cost involved in smart contract deployment and execution.

Fingerprint Templates		Contract Functions	Transaction Cost	Execution Cost	Execution Time (Average)
Features	**Method of Template Acquisition**				
Minutia based	BM	sendTemplate	665,608 ($2.63582)	583,664 ($2.31131)	2.69 s
		deleteTemplate	84,820 ($0.33589)	148,048 ($0.58628)	1.98 s
		getTemplate	49,776 ($0.19711)	28,504 ($0.11288)	1.45 s
	MP	sendTemplate	285,774 ($1.13167)	77,686 ($0.30763)	7.5 s
		deleteTemplate	231,473 ($0.91664)	441,354 ($1.74777)	5.60 s
		getTemplate	103,829 ($0.41117)	82,557 ($0.32692)	5.33 s
	MA	sendTemplate	3,468,447 ($13.73504)	3,113,287 ($12.32861)	11.13 s
		deleteTemplate	403,411 ($1.5975)	785,230 ($3.10951)	9.85 s
		getTemplate	167,268 ($0.66239)	145,996 ($0.57814)	11.38 s
Texture features	DWT	sendTemplate	243,266 ($0.96333)	202,218 ($0.80079)	2.08 s
		deleteTemplate	367,79 ($0.14565)	51,965 ($0.20578)	1.45 s
		getTemplate	32,081 ($0.12704)	10,809 ($0.04281)	1.40 s
Hashing of features	Hashing of DWT	sendHash	86,119 ($0.34104)	61,863 ($0.24499)	1 s
		deleteHash	18,468 ($0.07314)	16,743 ($0.06631)	1 s
		getHash	25,552 ($0.10118)	4280 ($0.01695)	1 s
smart-contract creation			338,033 ($1.33862)		1 s

Hashing of minutia based feature results at the same cost as hashing of the DWT method.

The sendTemplate/Hash, deleteTemplate/Hash, getTemplate/Hash function incurred transaction and execution costs, as illustrated in Figure 6. The execution time taken to send a template or hash by to the Ethereum network also varies. Figure 7 displays how our proposed system has high performance, because it just sends the hash to Ethereum when compared to that of the other templates from the different recognition techniques.

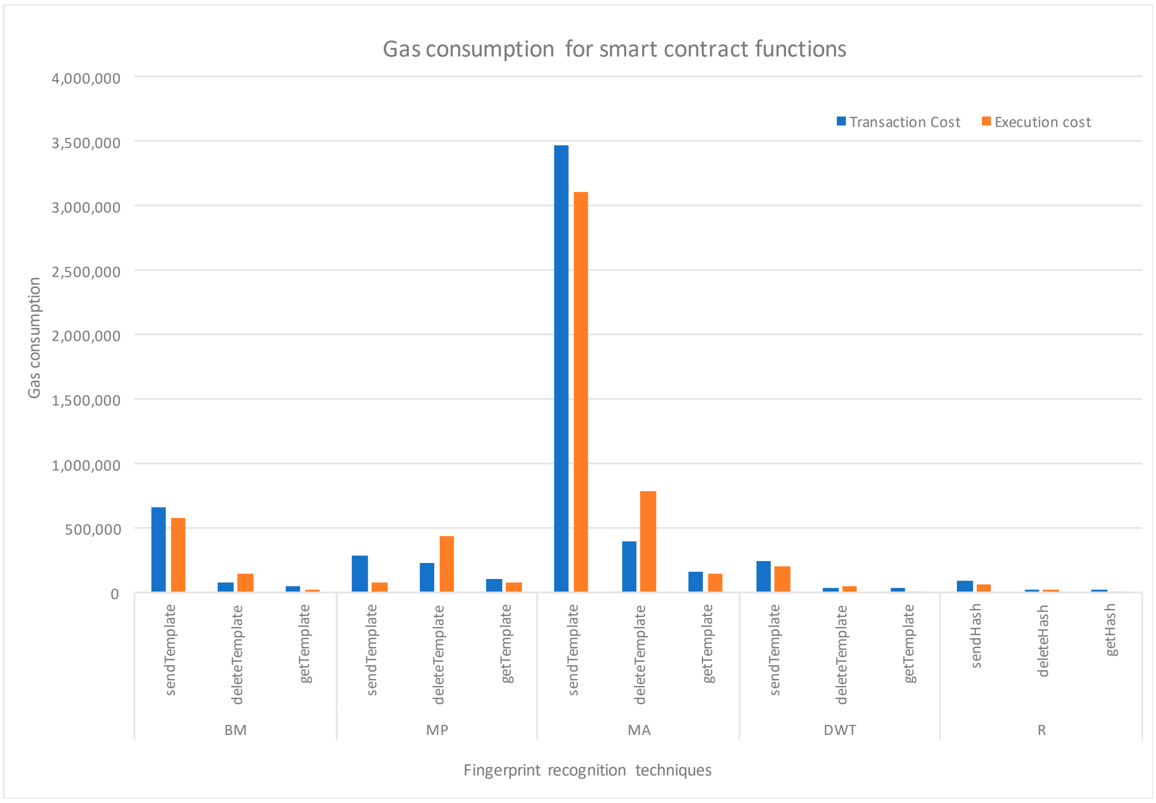

Figure 6. Gas consumption by smart contract.

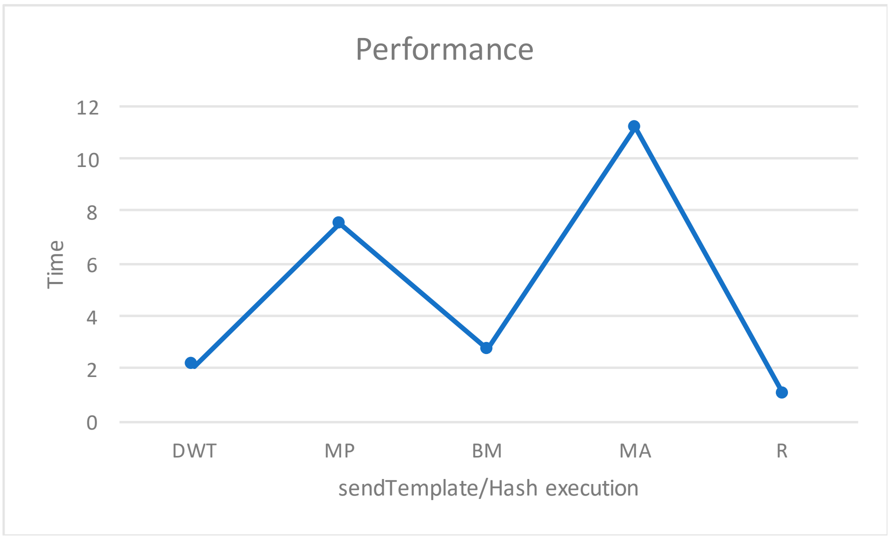

Figure 7. Performance of $\mathcal{R}$.

5.3. Security Analysis of the Improved System

The merits of our proposed system are laid out in this section. The traditional fingerprint data storage system is vulnerable to various attacks, and we analyze such attacks and how our proposed system solves them.

- Template database attack

The fingerprint database has a centralized data storage system, as shown in Figure 1, which results in the loss of data or user information theft. Our system proposes the use of IPFS, a distributed file system, and stores data by splitting it onto different nodes. The difficulty of attacking every node on the system is tremendous and encrypting the template makes it more difficult.

- Upload of fake template

Spoofing of a fingerprint system has long been an issue, and malicious attackers have resulted in this method to bypass the most secured systems. Therefore, we upgraded the system in order to eliminate this problem by utilizing the Ethereum network. The immutable feature of the Ethereum network ensures that all transactions cannot be changed. $\langle_g$ of every uploaded **E** is already written to the network.

- Channel interception

Middlemen or DBA turn to handle database or cloud storage for the traditional fingerprint system and, thus, given the authority to manipulate data or even sell user information. The Facebook-Cambridge Analytica scandal proves the disadvantage of centralized data storage or including middlemen. If a malicious node on the proposed system tries to alter the smart contract for benefits, it will result in being unsuccessful, because smart contracts are tamperproof once deployed.

5.4. Comparison of $\mathcal{R}$ with Dipti's System

In this section, we discuss the disadvantages in terms of differences of Dipti's system compared to our proposed system $\mathcal{R}$. Diptis proposed system is an improvement of the traditional fingerprint authentication system by using blockchain technology. Dipti's system starts enrollment by getting the Aadhar card number of the user and pushing it onto the blockchain to check if it exists. This procedure then continues with the side chain technique of processing the fingerprint and obtaining the users' information to store on the blockchain if it does not find an existing Aadhar card number. Our system

$\mathcal{R}$ improves the idea of the side chain, which, in Dipti's system, is another blockchain against the main blockchain. We used the distributed file system IPFS to store the encrypted fingerprint template **E**, and the returned hash is stored on the Ethereum network. The smart contract triggers the getHash function, which returns $\langle_g$. The system later uses $\langle_g$ for authentication Table 5 summarizes the difference between the two systems.

Table 5. Comparison of $\mathcal{R}$ and Dipti's system.

	Dipti's System	**Proposed System $\mathcal{R}$**
Setup	Side-chain	Distributed with IPFS
Encryption	No encryption of user data	User template is encrypted with AES **E**
Cost	Expensive from storing fingerprint features directly on the main chain	Cheaper by just storing $\langle$
Scalability & efficiency	Large files in the form fingerprint vector and user data increase the size of the system and become less efficient	Only using $\langle$ makes the system efficient

6. Conclusions

Blockchain technology is a widespread growth in the field of research and information security; hence, the proposed system is an appropriate replacement for the traditional fingerprint database. The introduction of the IPFS system, which is a cost-effective and distributed file system for storing and securing data, is implemented in our system. The proposed method is cost-efficient as compared to other decentralized biometric architectures. By comparing different fingerprint recognition templates, we displayed how cost-effective it is to store just the hash of the template on the Ethereum network. The decentralized structure also benefits the user in terms of security, accessibility, and trust. DBAs are no longer needed for this system. In summary, we proved that merging IPFS and public blockchain to secure encrypted fingerprint template is possible for economic and performance viewpoint, including two case studies with modernized methods and protocols in fingerprint template storage. Future extensions of this proposed system are to combine the "side chain" aspect of fingerprint processing into one unique authentication system. Another possible future extension is to derive a robust hashing function for different blockchain types.

Author Contributions: Conceptualization, M.A.A., and B.Y.; Data curation, N.C.; Formal analysis, N.C. and B.Y.; Investigation, M.A.A.; Methodology, M.A.A.; Project administration, B.Y.; Resources, B.Y.; Software, M.A.A.; Validation, M.A.A., and J.-S.P.; Writing—original draft, M.A.A.; Writing—review & editing, H.-M.Y. and B.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of China, grant number 61272432. It was also partially funded by MOE (Ministry of Education in China) Project of Humanities and Social Sciences, grant number 18YJAZH110.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

This section shows the class diagram Figure A1 of the storeHash smart contract and the corresponding commented sample code in solidity.

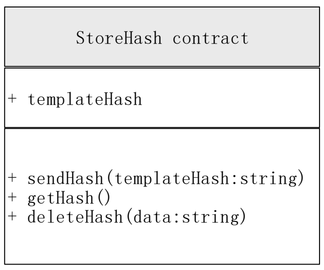

Figure A1. The storeHash class diagram.

```
//using specific solidity version
pragma solidity ^0.6.1;
contract uploadTemplate {
string templateHash;//data type of IPFS hash
//send the template hash to the Ethereum network
function sendHash(string _templateHash) public {
templateHash = _templateHash;
}
//return the template hash
function getHash() public view returns (string _template) {
return (templateHash);
}
//delete template hash on the Ethereum network
function deleteHash(string memory)public{
delete templateHash;
}
}
```

Appendix B

This section displays the sample code for submitting the encrypted template to IPFS, this represents the central part of the Algorithm 1. This code interacts with the storeHash smart contract from Appendix A.

```
onSubmit = async (event) => {
event.preventDefault();
//display user's metamask account address
const accounts = await web3.eth.getAccounts();
//obtain contract address from storehash.js
const ethAddress= await storehash.options.address;
this.setState({ethAddress});
//send Encrypted template to IPFS,return its hash
await ipfs.add(this.state.buffer, (err, templateHash) => {
console.log(err,templateHash);
//setState by setting templateHash to templateHash[0].hash
this.setState({ templateHash:templateHash[0].hash });
// call storeHash contract function "sendTemplate" and send IPFS hash to ethereum contract
//return the transaction hash from the ethereum contract
storehash.methods.sendTemplate(this.state.templateHash).send({
from: accounts[0]
}, (error, transactionHash) => {
console.log(transactionHash);
this.setState({transactionHash});
});
})
};
```

References

1. Makhdoom, I.; Abolhasan, M.; Abbas, H.; Ni, W. Blockchain's adoption in IoT: The challenges, and a way forward. *J. Netw. Comput. Appl.* **2019**, *125*, 251–279. [CrossRef]
2. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*; 2008; Available online: https://bitcoin.org/bitcoin.pdf (accessed on 1 June 2020).

3. Yli-Huumo, J.; Ko, D.; Choi, S.; Park, S.; Smolander, K. Where is current research on blockchain technology?—A systematic review. *PLoS ONE* **2016**, *11*, e0163477. [CrossRef] [PubMed]
4. Roberts, C. Biometric attack vectors and defences. *Comput. Secur.* **2007**, *26*, 14–25. [CrossRef]
5. 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*, 10. [CrossRef]
6. Nimje, R.; Bhalerao, D. Blockchain Based Electronic Voting System Using Biometric. In Proceedings of the International Conference on Sustainable Communication Networks and Application, Erode, India, 30–31 July 2019; pp. 746–754.
7. Pawade, D.; Sakhapara, A.; Badgujar, A.; Adepu, D.; Andrade, M. Secure Online Voting System Using Biometric and Blockchain. In *Data Management, Analytics and Innovation*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 93–110.
8. Odelu, V. IMBUA: Identity Management on Blockchain for Biometrics-Based User Authentication. In Proceedings of the International Congress on Blockchain and Applications, Ávila, Spain, 26–28 June 2019; pp. 1–10.
9. Jiang, X.; Yau, W.-Y. Fingerprint minutiae matching based on the local and global structures. In Proceedings of the 15th international conference on pattern recognition. ICPR-2000, Barcelona, Spain, 3–7 September 2000; pp. 1038–1041.
10. Khan, M.A. *Fingerprint Image Enhancement and Minutiae Extraction*; Cornell University: New York, NY, USA, 2011.
11. Kumar, L.R.; Kumar, S.S.; Prasad, J.R.; Rao, B.S.; Prakash, P.R. Fingerprint minutia match using bifurcation technique. *Int. J. Comput. Sci. Commun. Netw.* **2012**, *2*, 478–486.
12. Thaiyalnayaki, K.; Karim, S.S.A.; Parmar, P.V. Finger print recognition using discrete wavelet transform. *Int. J. Comput. Appl.* **2010**, *1*, 96–100.
13. Pawade, D.; Sakhapara, A.; Andrade, M.; Badgujar, A.; Adepu, D. Implementation of Fingerprint-Based Authentication System Using Blockchain. In *Soft Computing and Signal Processing*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 233–242.
14. Benet, J. IPFS-Content Addressed, Versioned, P2P File System. Whitepaper. 2014. Available online: https://www.researchgate.net/publication/263930348_IPFS_-_Content_Addressed_Versioned_P2P_File_System (accessed on 1 January 2020).
15. Delgado-Mohatar, O.; Fierrez, J.; Tolosana, R.; Vera-Rodriguez, R. Biometric Template Storage with Blockchain: A First Look into Cost and Performance Tradeoffs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–20 June 2019; pp. 2829–2837.
16. Christidis, K.; Devetsikiotis, M. Blockchains and smart contracts for the internet of things. *IEEE Access* **2016**, *4*, 2292–2303. [CrossRef]
17. Mohsin, A.; Zaidan, A.; Zaidan, B.; Albahri, O.; Albahri, A.; Alsalem, M.; Mohammed, K. Blockchain authentication of network applications: Taxonomy, classification, capabilities, open challenges, motivations, recommendations and future directions. *Comput. Stand. Interfaces* **2019**, *64*, 41–60. [CrossRef]
18. Buterin, V. Ethereum white paper. *Github Repos.* **2013**, *1*, 22–23.
19. Dannen, C. *Introducing Ethereum and Solidity*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 1.
20. Buterin, V.; Griffith, V. Casper the friendly finality gadget. *arXiv* **2017**, arXiv:1710.09437.
21. Nizamuddin, N.; Salah, K.; Azad, M.A.; Arshad, J.; Rehman, M. Decentralized document version control using ethereum blockchain and IPFS. *Comput. Electr. Eng.* **2019**, *76*, 183–197. [CrossRef]
22. Lee, S.H.; Yang, C.S. Fingernail analysis management system using microscopy sensor and blockchain technology. *Int. J. Distrib. Sens. Netw.* **2018**, *14*, 1550147718767044. [CrossRef]
23. Mohsin, A.; Zaidan, A.; Zaidan, B.; Albahri, O.; Albahri, A.; Alsalem, M.; Mohammed, K. Based Blockchain-PSO-AES techniques in finger vein biometrics: A novel verification secure framework for patient authentication. *Comput. Stand. Interfaces* **2019**, *66*, 103343. [CrossRef]
24. Nyaletey, E.; Parizi, R.M.; Zhang, Q.; Choo, K.-K.R. BlockIPFS-Blockchain-enabled Interplanetary File System for Forensic and Trusted Data Traceability. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July; pp. 18–25.
25. Hasan, S.S.; Sultan, N.H.; Barbhuiya, F.A. Cloud Data Provenance using IPFS and Blockchain Technology. In Proceedings of the Seventh International Workshop on Security in Cloud Computing, Auckland, New Zealand, 7–12 July 2019; pp. 5–12.

26. Jain, A.K.; Prabhakar, S.; Hong, L.; Pankanti, S. Filterbank-based fingerprint matching. *IEEE Trans. Image Process.* **2000**, *9*, 846–859. [CrossRef] [PubMed]
27. Kauba, C.; Uhl, A. Fingerprint recognition under the influence of image sensor ageing. *IET Biom.* **2017**, *6*, 245–255. [CrossRef]