Beccak: Combination of Blockchain and Keccak Algorithm for Improved Digital Signature Performace

Irfan Darmawan , Firman Maulana , Alam Rahmatulloh * , Rohmat Gunawan

Posted Date: 5 June 2024

doi: 10.20944/preprints202406.0182.v1

Keywords: architecture; blockchain; digital certificate; digital signature; Keccak256 Hash

Article

Beccak: Combination of Blockchain and Keccak Algorithm for Improved Digital Signature Performace

Irfan Darmawan [1], Firman Maulana [2], Alam Rahmatulloh [2,*] and Rohmat Gunawan [2]

[1] Department of Information Systems, Faculty of Industrial Engineering, Bandung, (Indonesia
[2] Department of Informatics, Faculty of Engineering, Siliwangi University, Tasikmalaya (Indonesia)
* Correspondence: alam@unsil.ac.id

Abstract: Data integrity and security are among the most essential things in this digital era performance. Digital certificates can be one solution to this problem. This research will develop the architecture of digital certificate verification and validation using blockchain. The blockchain concept for digital certificates uses three stages: requirements planning, workshop design, and implementation. Digital certificate files will be tested based on a scenario in which there could be any changes to those files, such as compressing files, changing data of files, inserting images in the file, and rotating files. This system architecture will detect any changes that could occur to the digital certificate file by using a digital signature, Keccak256 hash algorithm, and blockchain. This research shows that change detection accuracy for digital certificate files is 100%. However, there is a digital signature nature that could reduce the accuracy of change detection, and several solutions can be resolved. The average time to generate a signature using blockchain technology is 2.11 seconds, faster than 3.23 seconds when it does not use blockchain. The average time to validate a signature is 2.11 seconds without using blockchain, which is reduced to 0.22 seconds using blockchain technology. This research concludes that the architecture for digital certificate verification and validation using blockchain technology has successfully improved the security of digital certificates. Testing showed that even minor changes to a digital certificate could affect the signature, ensuring the certificate's integrity. For further research, evaluating the performance of various hash algorithms within the blockchain network is recommended.

Keywords: architecture; blockchain; digital certificate; digital signature; Keccak256 Hash

1. Introduction

As we navigate the digital era, data integrity and security emerge as paramount concerns. Digital certificates, for instance, present a potential solution to this challenge (Lorien & Wellem, 2021). These electronic documents serve as a repository of the owner's identity information (Lorien & Wellem, 2021). In light of this, our research endeavors to develop an architecture that can effectively verify and validate the integrity of digital certificates, a crucial step in ensuring the security of our digital landscape.

While digital signatures have been proposed as a solution (Somsuk & Thakong, 2020), there are still significant issues surrounding the authenticity of documents with digital signatures (Yuniati & Sidiq, 2020). Presently, there is no robust guarantee of the authenticity and integrity of documents that use digital signatures, leaving them vulnerable to forgery or data manipulation. This lack of assurance can potentially erode users' trust in the validity of digital documents (Yuniati & Sidiq, 2020). Hence, there is a pressing need for a more secure solution.

Digital signatures are a public-key primitive in message authentication (Nadzifarin & Asmunin, 2022). In the physical world, handwritten signatures are common on written or typewritten messages, serving to associate the signatory with the transmitted message. Digital signatures similarly act as a technique that links an individual or entity to digital data. The recipient or a third party can independently verify this association.

 doi:10.20944/preprints202406.0182.v1

The digital signature model illustrated in Figure 1 utilizes a set of private keys, where the private key is employed for signing purposes and the key for verification. The process commences as a user inputs data into a hash function to produce a hash value. Subsequently, this hash value is signed using the key, resulting in a signature accompanying the data to the verifier. The verifier integrates the signature and the public key into a verification algorithm, executing the hash function on the data to derive another hash value. Suppose this new hash value corresponds with the output from the verification algorithm. In that case, it confirms the validity of the signature, ensuring that the signer cannot refute having signed the document at a time.

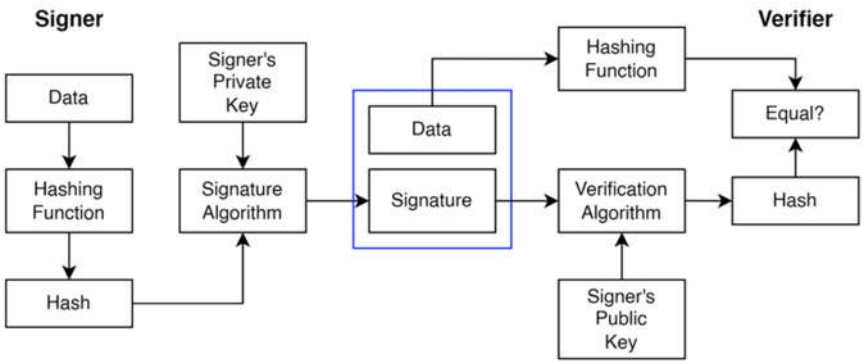

Figure 1. Digital Signature Model (Sasi et al., 2023.)

Relying on signatures may not be sufficient to ensure digital certificates; however, blockchain technology presents a solution (Fang et al., 2020; Sowmiya et al., 2021; Zou & Zeng, 2023). Blockchain operates as a distributed data storage mechanism that employs cryptography to record transactions in interconnected blocks (Zheng et al., 2017). Serving as infrastructure, blockchain provides an auditable environment particularly suited for managing and safeguarding digital certificates (Gayathiri et al., 2020).

The features of networks consist of (*Keccak-256 Online Computing - StrErr.Com*, n.d.; Zheng et al., 2017):

1. Decentralization: Traditional centralized transaction systems need validation from an authority, which results in costs and performance limitations due to server overload. On the other hand, blockchain technology removes the necessity for a third-party authority for validation by using consensus algorithms to uphold data consistency across the network.
2. Persistency: Transactions can be quickly validated, and miners do not accept invalid transactions. Once included in the blockchain, it is challenging to delete or reverse transactions. Blocks containing invalid transactions can be quickly detected.
3. Anonymity: Users can interact with the blockchain using generated addresses that do not reveal their identity. However, due to intrinsic limitations, blockchain cannot guarantee perfect privacy.
4. Auditability: Bitcoin blockchain stores user balance data using the Unspent Transaction Output (UTXO) model. Every transaction must refer to several previously unspent transactions, ensuring easy verification and tracking when the transaction status changes from unspent to spent.

A blockchain works similarly to a spreadsheet, which contains a record of transactions that are then widely replicated across a computer network. The goal of this network design is to have the worksheet updated regularly. Each person can record their transactions in their spreadsheet, as shown in this image because blockchain requires no intermediary. Using internal algorithms in the blockchain enables network consensus every time new data is added (Laurence, 2023).

This research adds a hash algorithm to the digital certificate before digital acknowledgment and delivery to the blockchain network. A mathematical function known as a hash algorithm converts input data, or messages, into a set of hash values that uniquely represent that input (Santoso et al., 2019). Integrating digital signatures and hash algorithms with blockchain can increase data security

 doi:10.20944/preprints202406.0182.v1

(Saputra & Nasution, 2019). Improving the security of digital certificates can be done with the verification and validation process on the blockchain network [4]. Several hash algorithms are commonly used on blockchain, namely SHA256, Keccak256, Blake2, and others (Alshaikhli et al., 2012). Keccak256 is a faster hash algorithm than other algorithms (Kuznetsov et al., 2021). Keccak256 is a SHA3 family and a cryptographic function. Keccak256 calculates the input into a fixed-size output of 32 bytes. The hash function can only be used one way and cannot be reversed. Given a string input like "Namaste Duniya," the Keccak256 hash function converts it into a unique hash value. For example:

```
"Namaste Duniya" -> keccak-256(hash function) ->
8a0fe4fd16bb35fbecde2e774008fb7f92a8568a680f3fa93d0948bcfbf68dc3
"namaste duniya" -> keccak-256(hash function) ->
544ff8f09d34fdfbfa5a8dcb9d7d57deb6c862026cb8b655cfd7bf9c192e4d21
```

The difference between these two outputs demonstrates the sensitivity of hash functions, where even minor changes can result in vastly different hash digests.

The issue of forged diplomas or transcripts is a significant problem in higher education institutions. To combat this fraudulent activity, the Higher Education Service Agency has implemented the Online Diploma Verification System (SIVIL) and the National Diploma Numbering System (PIN). However, these systems are vulnerable to SQL injection attacks due to their centralized nature. Our proposed solution, which advocates for replacing the SIVIL and PIN system with a distributed storage system using the InterPlanetary File System (IPFS) and smart contract blockchain for diploma or transcript hashing (Danil Muis et al., n.d.), could revolutionize how we verify educational credentials.

The next research study delves into the innovative use of blockchain and cryptography in a prototype electronic voting system. The paper details the protection of the integrity and security related to the voting of software and hardware, addressing the challenges in the application of blockchain concerning the Electronic Voting System. The proposed solution offers security, transparency, anonymity, and verification of this prototype system. This is achieved through blockchain to record the votes on the blockchain and then use the cryptography algorithm to check the votes information and the result (Shidqi, n.d.).

The following study deals with the importance of the validity of an educational certificate. The challenge is to verify this certificate quickly. Here, the options are to either use the help of the registrar's office or go through the online verification tutorials. However, the proposed idea using blockchain to store certificates, convert the paper certificates to digital format using the cryptographic algorithm (Skipjack, Advanced Encryption Standard, Secure Hash Algorithm, Rivest–Shamir–Adleman, Triple-DES, and Idea), and provide the validation via mobile application demonstrates that blockchain is the key to showcasing a tremendous increase in the security and integrity in a digital certificate (Gayathiri et al., 2020).

The final study explores the use of blockchain technology in Internet of Things (IoT) applications. A hybrid HW/SW architecture is proposed to address the main issues of power consumption and execution time of IoT applications. The results demonstrate a significant decrease in execution time and a significant drop in power consumption, showcasing the potential of blockchain technology (Frikha et al., 2021).

The application of digital signatures and the Keccak256 hash algorithm into the blockchain network on digital certificates is the novel focus of this research. Our main goal is to enhance security and maintain the integrity of digital certificates. The unique aspect of our research lies in the use of a combination of digital signatures and the Keccak256 hash algorithm, along with the integration of blockchain technology, to create a more secure digital certificate verification and validation system. Furthermore, we have conducted special tests on files, including compression, data modification, adding images, and file rotation, to ensure the robustness of our system.

2. Methodology

The research process to be applied is illustrated in Figure 2.

 doi:10.20944/preprints202406.0182.v1

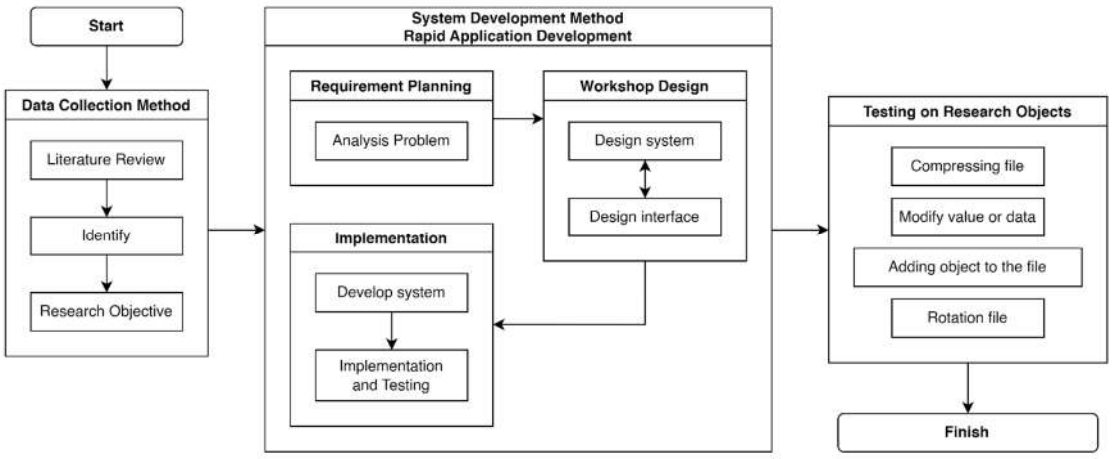

Figure 2. Research Stages.

2.1. Data Collection Method

This research began with the application of Data Collec-tion Methods. The first step is literature study, which is carried out to obtain information that is relevant to the research topic. Next, Problem or Need Identification is carried out to determine the issue or need that this re-search wants to address. The next stage is the formula-tion of the hypothesis or research objectives which is the basis for the research direction and the framework for evaluating the expected results. This entire process aims to ensure that the research has a clear focus and is based on comprehensive information.

2.2. System Development Method

The design of the digital certificate verification and validation system architecture using blockchain technology adopts the Rapid Application Development (RAD) system development approach. RAD consists of three main stages (Susilo et al., 2023):

1. Requirement Planning

This stage involves two main steps:

- Problem Needs Analysis, the initial stage of the research where the aims, objectives, and goals of the application are analyzed, which will then be discussed with the supervising lecturer.
- Problem Definition, where the problems to be solved are established. The issues are identified after completing the Problem Needs Analysis.

2. Workshop Design

The design stage involves system design and enhancement if there are deficiencies in the initial design. The system design stage comprises two main parts:

- System Design, which involves creating a verification and validation system that consists of two main elements: a web application as the client and a smart contract that contains business logic.
- Interface Design, in this stage, the system's interface is designed. The design of the system's interface uses Figma as a design tool

3. Implementation

The system is programmed using MySQL for the database, ReactJS for the frontend, Laravel for the backend, and JavaScript and Solidity for smart contracts. The system is then tested using black-box testing. The aim of this testing is to ensure that the system's functions, inputs, and outputs meet the desired specifications.

2.3. Testing on Research Object

The research concludes with Testing on Research Objects to ensure the system functions according to the specified criteria and meets the research objectives. The testing includes:

1. Compressing digital certificate files.
2. Modifying values or data in digital certificate files.
3. Adding images to digital certificate files.

 doi:10.20944/preprints202406.0182.v1

4. Rotating or flipping the contents of digital certificate files.

3. Result and Discussion

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn.

3.1. Requirement Planning

The Requirement Planning stage involves two crucial aspects:

1. Problem Needs Analysis: From existing research, it is clear that most applications require security and integrity of data managed by digital certificates. Due to the benefits of digital signature technology in proving integrity and the use of blockchain in proving integrity as exposed in existing research. Blockchain stores and records data in a decentralized manner, using immutable ledgers and powerful hash functions. Computational trust is established through checking and balancing across every node in the network, eliminating the need for a third party. This study aims to develop solutions that strengthen the security and integrity of digital certificates, focusing on the application of digital signatures within a blockchain network and the use of the Keccak256 hash algorithm.
2. Problem Definition: By identifying the issues at hand that could be solved by implementing the system under development, this phase focuses on stating the problem at hand. Since the central idea of this study is to design a verification and validation system for digital certificates using the Ethereum blockchain platform and the Keccak256 hash algorithm, defining the problem would be creating a system that will safeguard digital certificates' security and integrity in an ever-changing technological world.

3.2. Workshop Design

3.2.1. System Design Stage

Based on Figure 3, users can register, enter the system (login), view the sign document history, sign document, and verify document. Figure 4 is a general overview of the architecture that will be created. This architecture was created using ReactJS as the frontend, Laravel as the backend, and MySQL as the database. To communicate with the blockchain use the web3js library and the metamask extension.

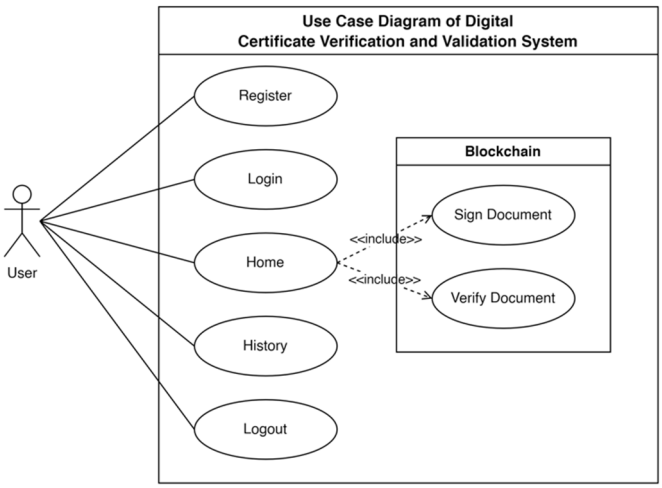

Figure 3. Use Case Diagram for Digital Certificate Verification and Validation System.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2024 doi:10.20944/preprints202406.0182.v1

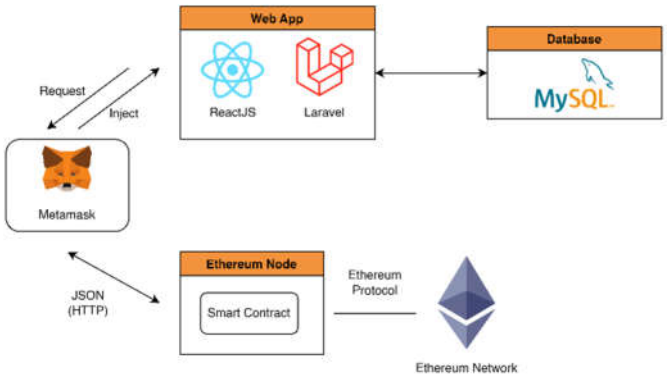

Figure 4. System Overview.

Figure 5 is the flow of the verification and validation system. At the signing stage, the input file from the user is then hashed using the keccak256 algorithm and the resulting hash is submitted to the blockchain network. At the verification stage, user input the file and then hash it and the hash result is compared with the signature from the blockchain network. If the hash is the same as the signature then there is no change in the file and if the hash is not the same as the signature then there is a change in the file.

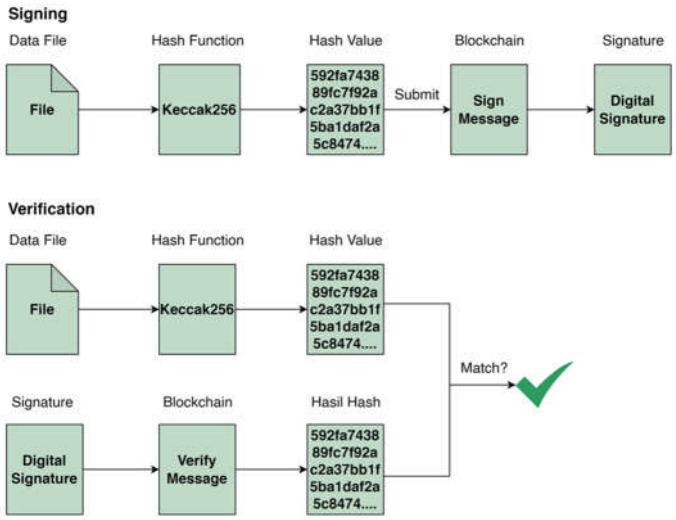

Figure 5. Digital Certificate Verification and Validation System Workflow.

3.2.2. Keccak Function and Digital Signature Blockchain

1. Sponge Contruction

Sponge construction is a mode of operation that builds the functionality of the sponge (Bertoni et al., 2011). The sponge function is a generalization of the hash function. As appeared in Figure 8, the sponge design operates with the bit states $b = r + c$, and all initial states are set to the initial value of '0'. During the absorbing phase, the message M is padded to a length that is a multiple of r. The layered input message is partitioned into several R-bit message blocks. Each time a state captures a message block, the first r bits of the state are XOR-ed with the R-bit message block. The states are at that point controlled with Keccak-f stages. During the squeezing phase, the state extracts each r bits by removing the first r bits from the state until the output length is equal to or greater than the desired length. Similar to the absorption phase, the state is manipulated via Keccak-f permutations each time producing an R-bit output. Finally, cut the output to the required length to get the essence.

2. Keccak-f Permutation

Although the state variable b can be chosen from {25, 50, 100, 200, 400, 800, 1600}, NIST has chosen the value 1600 for b as the SHA-3 standard. This article focuses on the case when b = 1600. As

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2024 doi:10.20944/preprints202406.0182.v1

shown in Figure 6, a 1600-bit state can be represented as a 5 × 5 64-bit trace. States can be symbolized as Ax, y, z. where 0 ≤ x, y ≤ 4, 0 ≤ z ≤ 63. The Keccak-f [1600] permutation consists of 24 spin functions that differ only in their spin-dependent constants. The round function R has 5 steps $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$:

$$\theta : A_{x,y,z} = A_{x,y,z} \oplus M_{i=0\sim4}(A_{x-1,i,z} \oplus A_{x+1,i,z-1}) \quad (1)$$

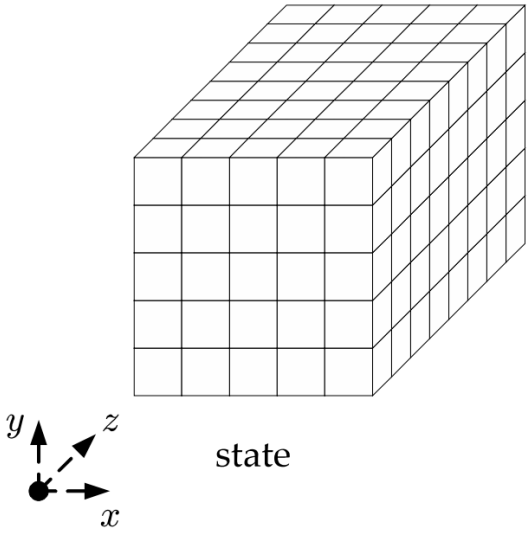

Figure 6. Keccak-f State.

Formula 1 is Theta (θ), at this stage a bitwise XOR operation is carried out in the 5x5 matrix column. This stage calculates the XOR on the elements of each column with the previous column which has been rotated.

$$\rho : A_{x,y,z} = A_{x,y,(z-rx,y)} \quad (2)$$

Formula 2 is Rho (ϱ), at this stage a bit rotation is carried out on one of the matrix elements. The matrix elements are rotated according to the number of indexes of the element's position.

$$\pi : A_{x,y,z} = A_{x+3y,x,z} \quad (3)$$

Formula 3 is Pi (π), at this stage a permutation of the matrix elements is carried out. This stage takes each element and moves it to a new point.

$$\chi : A_{x,y,z} = A_{x,y,z} \oplus (A_{x+1,y,z} \oplus 1) \cdot A_{x+2,y,z} \quad (4)$$

Formula 4 Chi (χ), at this stage an XOR operation is carried out for each row of the matrix between each element and the next two elements in the same row.

$$\iota : A_{0,0,z} = A_{0,0,z} \oplus RC_z \quad (5)$$

The 5 Iota (ι) formula, at this stage, is added to the matrix of bitwise constants. Each constant depends on the round being run in the algorithm.

The symbol "⊕" in formulas 1, 4, and 5 indicates bitwise XOR, while "·" denotes bitwise AND. The variables x and y are calculated modulo 5, and z is calculated modulo 64. rx, y are the constants listed in Table 1, and RCz is a rotation-dependent constant.

Table 1. Rotation Offsets.

	X = 0	**X = 1**	**X = 2**	**X = 3**	**X = 4**
Y = 0	0	1	62	28	27
Y = 1	36	44	6	55	20

 doi:10.20944/preprints202406.0182.v1

Y = 2	3	10	43	25	39
Y = 3	41	45	15	21	8
Y = 4	18	2	61	56	14
Y = 0	0	1	62	28	27

3. SHA-3 (Keccak) Standard

Table 1 shows the rotation offsets for the Keccak algorithm, designed to ensure effective diffusion during the hashing process. In cryptography, diffusion is a critical design principle aimed at ensuring that even small changes in input lead to significant changes in output, known as the avalanche effect. This increases security by making it difficult for attackers to predict hash outcomes or find two inputs that yield the same hash value.

There are four versions of SHA-3 standardized by NIST (Foti, 2015). The parameters for these versions are $r = 1600 - 2\ell$ and $c = 2\ell$, where $\ell \in \{224, 256, 384, 512\}$. The main distinction between SHA-3 and Keccak is the padding rules. Keccak and SHA-3 messages pad M with "10∗1" and "0110∗1" respectively.

4. Digital Signature Blockchain

According to Figure 7, the Keccak hash (digest value) is then signed using a digital signature algorithm, where:

$$r = (gk \bmod p) \bmod q \tag{6}$$

Based on formula 6, three global parameters are p, q, and g. The parameter p is a prime number where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L as the bit length between 512 and 1024 in 64-bit increments. Parameter q is an N-bit prime dividing p where $2^{N-1} < q < 2^N$. Parameter g is chosen using the formula $h^{(p-1)/q} \bmod p$ where h is an integer between $1 < h < (p-1)$.

$$s = [k - 1\,(H(M) + xr)] \bmod q \tag{7}$$

In formula 7, creating a digital signature involves calculating two quantities, r and s, which depend on the public key components (p, q, g), the user's private key (x), the message's hash code $H(M)$, and an additional integer k which should be randomly or pseudorandomly generated and unique for each signing (Stallings, 2013).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2024 doi:10.20944/preprints202406.0182.v1

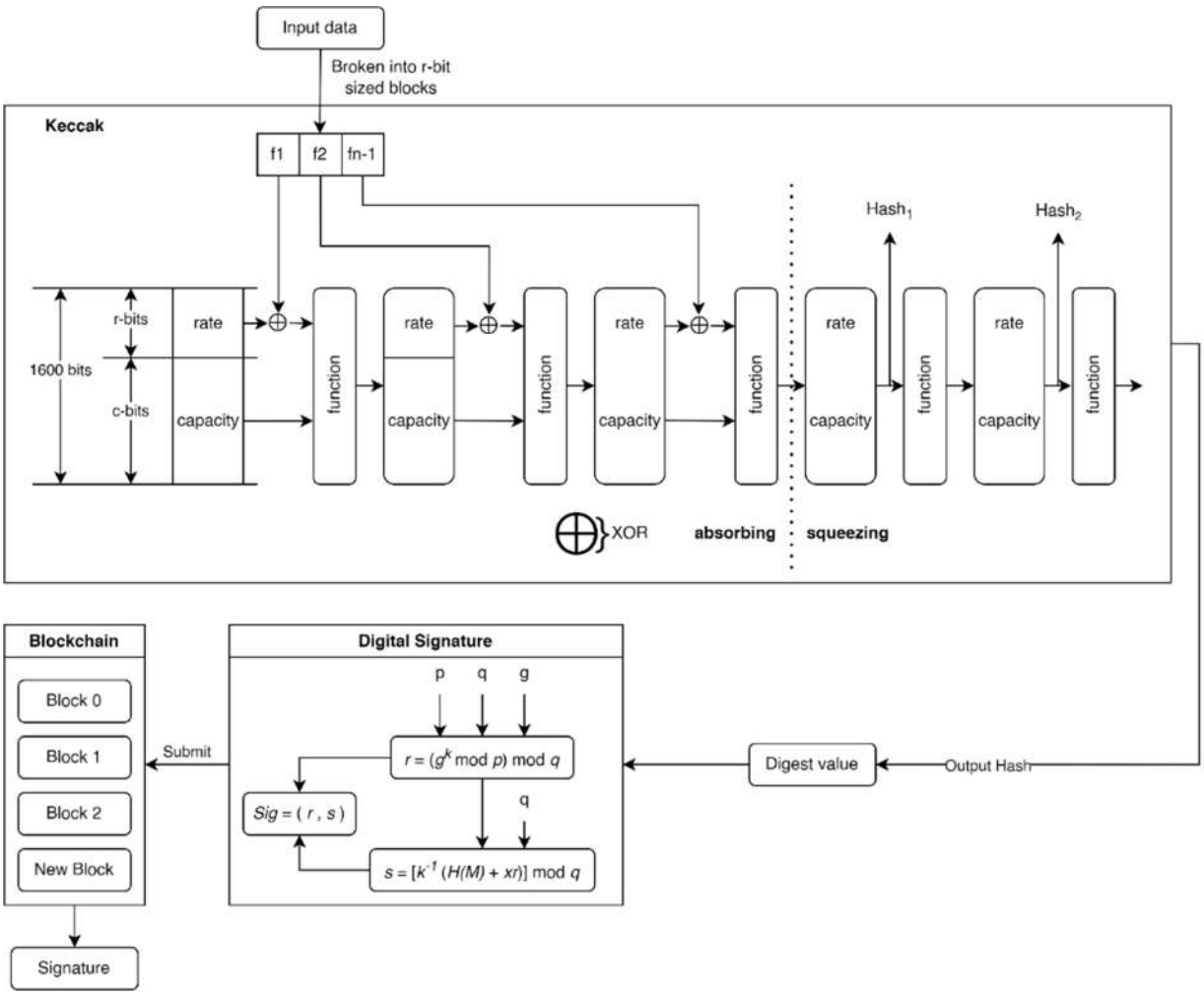

Figure 7. Keccak Hash and Digital Signature Blockchain.

The result of the digital signature is then submitted to the blockchain by adding a new block. Subsequently, the blockchain provides the signature as a marker, which will be used to verify the file on the blockchain network.

3.2.3. SHA Comparison

There are three SHA families: SHA1, SHA2, and SHA3 (Keccak). This section will present a comparison of the three families.

Table 2 is a comparison of several algorithms, namely SHA1, SHA256, and SHA3-256 (Keccak). The output of SHA-1 is 160 bits and the internal state size is 160 bits (5 x 32) with a block size of 512 bits and a maximum message length of (264 - 1) bits. The output of SHA256 is 256 bits and the internal state size is 256 bits (8 x 32) with a block size of 512. The output of Keccak256 is 256 bits and the internal state size is 1600 bits (5 x 5 x 64) with a block size of 1088 bits. The advantage of Keccak is that there is no limit to the maximum message length, aka unlimited.

Table 2. SHA Comparison.

Algoritma	Output Length (bit)	Internal state size (bit)	Block size (bit)	Maximum message length (bit)
SHA-1	160	160 (5 × 32)	512	$2^{64} - 1$
SHA256	256	256 (8 × 32)	512	$2^{64} - 1$

 doi:10.20944/preprints202406.0182.v1

SHA3-256 (Keccak)	256	1600 (5 × 5 × 64)	1088	Unlimited

Figure 8 shows three hash codes: SHA-1, SHA256, and SHA3-256 (Keccak). This looks at their power in stopping collisions, preimage attacks, and second preimage attacks in a matter of bits. SHA-1 and SHA256 have 160-bit and 256-bit power for collision and preimage attacks, respectively. On the other hand, there was an increase in Keccak256 in the preimage resistance section with 1600bit and the second preimage resistance, which was 512bit.

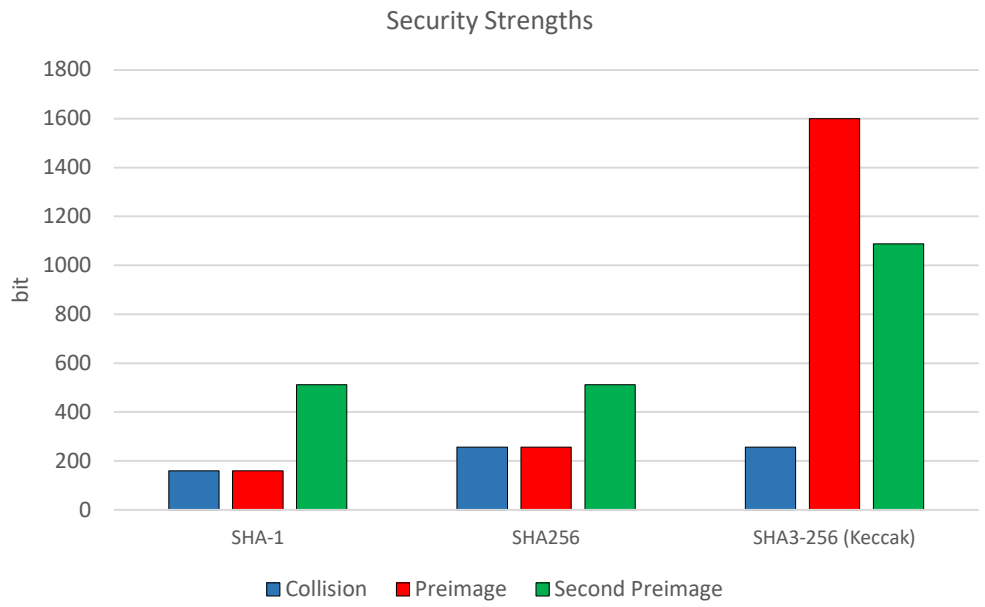

Figure 8. Security Strengths.

Table 3 shows a comparison of the average time required to generate and validate digital signatures using various algorithms. Iteration was carried out 10 times to get the time rates. In the table, "DS+SHA-1" takes an average of 2.04 seconds for signature generation and 1.40 seconds for validation. "DS+SHA256" takes a little longer, namely 2.36 seconds for generation and 1.49 seconds for validation. Meanwhile, "DS + SHA3-256 (Keccak)" takes the longest time to create a signature with an average of 3.23 seconds, but is faster in validation at 2.11 seconds compared to the previous two algorithms without blockchain integration. When the blockchain is integrated, "DS + SHA1 + Blockchain" takes 2.40 seconds for generation but significantly reduces validation time to only 0.19 seconds; Similarly, "DS + SHA256 + Blockchain" takes slightly less time to build (2.31 seconds) and an equally impressive reduction in validation time (0.20 seconds). Lastly, "DS + SHA3-256 (Keccak) + Blockchain" offers balanced performance with a build time of 2.25 seconds and a validation time of only 0.22 seconds.

Table 3. Performance of SHA.

Algoritma	Average time to generate signature (seconds)	Average time to validate signature (seconds)
DS + SHA-1	2.04	1.40
DS + SHA256	2.36	1.49
DS + SHA3-256 (Keccak)	3.23	2.11
DS + SHA1 + Blockchain	2.40	0.19
DS + SHA256 + Blockchain	2.31	0.20
DS + SHA3-256 (Keccak) + Blockchain	2.25	0.22

 doi:10.20944/preprints202406.0182.v1

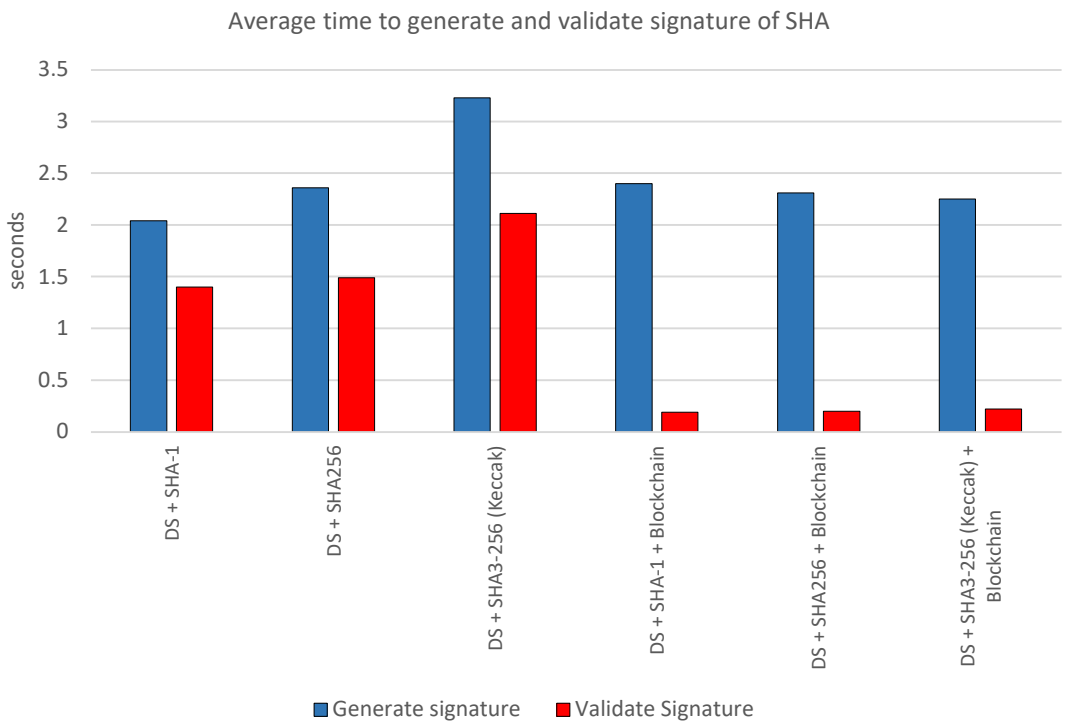

Figure 9. Average time to generate and validate of SHA.

Figure 9 is a diagram of the SHA performance comparison. This happens because blockchain technology allows data to be stored and accessed in a decentralized manner. With data that has been hashed using blockchain technology, the validation process becomes faster because the system only needs to check for consistency and suitability.

3.2.4. Interface Design Stage

The interface design stage in this research involves five UI/UX design planes (Garrett, 2011):

- Strategy Plane: The initial stage where goals, vision, and strategy for the digital certificate verification and validation system are created.
- Scope Plane: Defines the project scope by identifying key features, critical functions, and constraints.
- Structure Plane: The stage where the information structure and user interaction design are created.
- Skeleton Plane: Involves creating wireframes or simple prototypes to determine the placement of UI elements and the general layout of pages.
- Surface Plane: Develops a more detailed visual design with the choice of colors, fonts, graphics, and other design elements.

3.3. Implementation

3.3.1. Implementation Smart Contract

1. Handle Sign Document

The signBlockchain() function is a procedure used to digitally sign documents utilizing Ethereum blockchain and Metamask.

- It starts by initializing a Web3 object using the Metamask extension installed on the browser, done by calling the Web3(window.ethereum) constructor.
- The function then requests user permission to connect to Metamask and access the linked Ethereum account through the eth_requestAccounts method. Once granted, the Ethereum account address is stored in the account variable.

 doi:10.20944/preprints202406.0182.v1

- Next, it inputs and downloads the content of the file to be signed. After downloading, it calculates the file's hash using the keccak256() function, storing the result in the documentHash variable.
- The digital signature process uses the previously approved Ethereum account, involving the account's private key or calling the appropriate function from Ethereum smart contracts. The signature result is stored in the signature variable.
- Finally, the application data is updated with blockchain-related information, adding the blockchain document hash and the blockchain signature to the existing data structure.

```
  async function signBlockchain() {
  // Initialize Web3 with MetaMask Provider
  const web3 = new Web3(window.ethereum);

  // Request Permission to Connect to MetaMask
  const accounts = await window.ethereum.request({
  method: "eth_requestAccounts",
  });
  const account = accounts [0];

  // Download File
  const fileContent = await downloadFile(data.filePreview);
  const documentHash = keccak256(fileContent),

  // Sign Document
  const signature = await signDocument(web3, documentHash,
account);

  setData({
  ...data,
  blockchain_document_hash: documentHash,
  blockchain_signature: signature,
  });
•   }
```

2. Handle Verify Document

The documentVerification(dokumen) function is an asynchronous procedure used to digitally verify documents. This function takes a document parameter, representing the content of the document to be verified.

- It initializes a Web3 object using the installed Metamask extension in the browser, following the same approach as signBlockchain().
- It calculates the document's hash using the calculateFileHash(dokumen) function and stores the result in the documentHash variable.
- The function then attempts to find the document based on the hash in the blockchain by calling getDokumenByBlockchainHash(documentHash), with the result stored in the data variable.
- If the document search is successful, it verifies the signature by extracting the signature from the blockchain and validating it with verifySignature() using document hash, signature, and Web3 object as parameters.
- After signature verification, document verification follows. The signer's address from the verification is compared to the address recorded on the blockchain; if they match, the document

 doi:10.20944/preprints202406.0182.v1

is considered verified. The showResult and isVerified variables are set based on the verification outcome.

- If there are errors during verification or document search, error messages are logged to the console, and showResult and isVerified variables are set to indicate unsuccessful verification.

```
  async function documentVerification(dokumen) {
  const web3 = new Web3(window.ethereum);

  // result hash keccak256
  const documentHash = await calculateFileHash(dokumen);

  setLoading(true);
  try {
  // search document by hash
  const { data } = await
getDokumenByBlockchainHash(documentHash);

  const signature = data.blockchain_signature;
  const address = data.ethereum_address.address;

  // verify signature
  const signerAddress = verifySignature(
  web3,
  documentHash,
  signature
  );

  // verify address
  if (signerAddress.toLowerCase() === address.toLowerCase())
{
  setShowResult(true);
  setIsVerified(true);
  setDokumenData(data);
  } else {
  setShowResult(true);
  setIsVerified(false);
  }
  } catch (error) {
  console.log(error.response.data);
  setShowResult(true);
  setIsVerified(false);
  }
  setLoading(false);
  }
```

3.3.2. Application Outcomes

 doi:10.20944/preprints202406.0182.v1

1. Sign Document Page

Figure 10 is a display of the sign document page. Users are required to install the Metamask extension first, otherwise a warning will appear. Then the user can input the digital certificate file which will be digitally signed. Files uploaded to perform digital signatures only have the pdf extension.

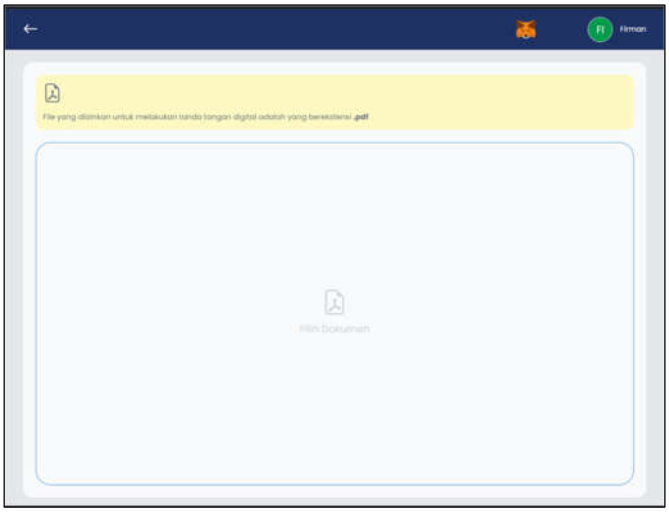

Figure 10. Sign Document Page Layout.

Figure 11 displays the Sign Document page after the user uploads the file. Here, users can view the uploaded certificate and set the QRCode coordinates before proceeding with the digital signature process. Additionally, users can reset the file by pressing the "Reset File" button if needed.

Figure 11. Sign Document Process Layout.

2. Verify Document Page

Figure 12 shows the Verify Document page interface after inputting the digital certificate file and submitting it. The system will display a blockchain signature if the document is valid. There is data displayed including who did the signing, ethereum address, document hash, and signature. Then the file is declared valid if it contains this data.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2024 doi:10.20944/preprints202406.0182.v1

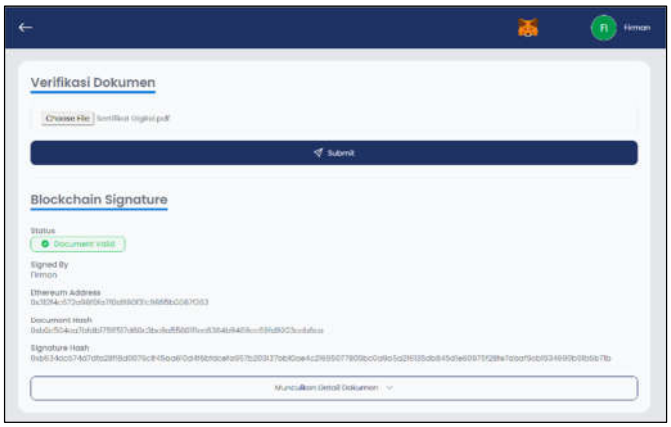

Figure 12. Valid Verify Document Page Layout.

Figure 12 hows the display if the digital certificate file is invalid. The system will detect changes and will display the message "Document not verified / invalid" if the file entered is invalid.

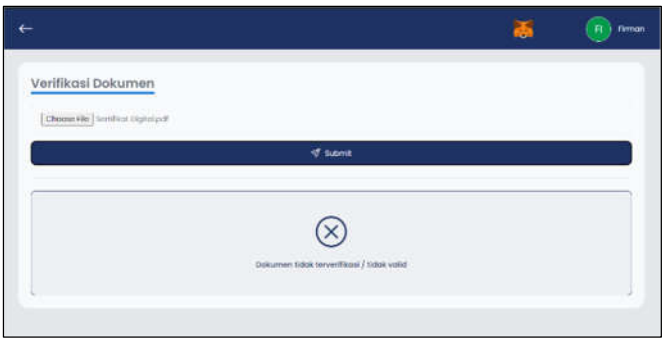

Figure 13. Invalid Verify Document Page Layout.

3. History Page

Figure 13 is a history page display which will display the user's digital signature history. There is a search feature based on subject matter to make it easier for users. Apart from that, users can delete history data.

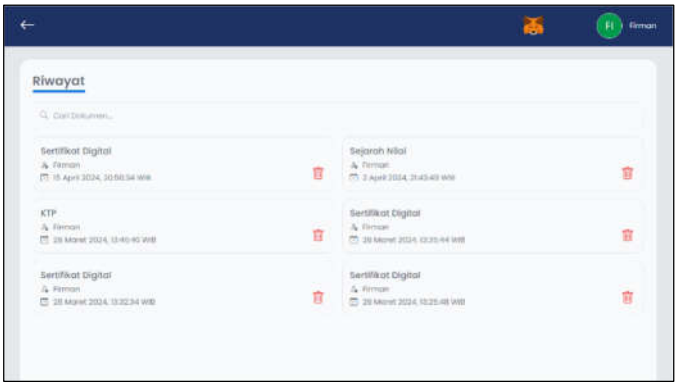

Figure 14. History Page Layout.

According to Table 4, the system testing using the blackbox testing method has been conducted. The results from several testing stages, starting from the register page, login page, sign document page, verify document page, and history page, were declared successful.

Table 4. System Testing.

Test Case	Testing Procedure	Expected Output	Result	Conclusion

 doi:10.20944/preprints202406.0182.v1

Register Page	Open the register page and register	Displays the register page and registers successfully	Successful	Valid
Login Page	Open the login page and log in	Displays the login page and logs in successfully	Successful	Valid
Sign Document Page	Open the sign document page and sign a document	Displays the sign document page and successfully signs the document	Successful	Valid
Verify Document Page	Open the verify document page and verify a document	Displays the verify document page and successfully verifies the document	Successful	Valid
History Page	Open the history page and display digital signature history	Displays the history page and successfully displays digital signature history	Successful	Valid

3.3.3. Testing

3.4. Testing on Research Objects

Parameter	Initial File	Manipulated File	Signature Value of Initial File	Signature Value of Manipulated File	Result	Conclusion
Compressing a digital certificate file			0xb428495f1fc649dd8235bfc12f9e092362a378c0cc96c4d51802a97d89ad0e1d279508ed6f0298eb46c1944f2936132ee2ca1d7f87c6f60e95a1b5597406049131c	0x54e7db1438c841ab4eb23ef0cb006041bb447699375f725252b8370424e88529257c43744583f79f62c9a74fa7a31d7d93e59ed1b59a5adc903b6dc83afc258701c	Detected	Successfully detected changes in the manipulated file
Modifying the values or data in a digital certificate file (by converting it to Word format and then saving it as a PDF)			0xb428495f1fc649dd8235bfc12f9e092362a378c0cc96c4d51802a9b2e89324b047e77d89ad0e1d279508ed6f0298eb12dea954373ef646c1944f2936133cf53d23ee2da1ee2ca1d7f87c6f57c0265d33e3e660e95a1b55974807812d0205d206049131c	0x3ee3dd5bac857158abf59885a332b5d98150d9f0616dd7a6f828	Detected	Successfully detected changes in the manipulated file
Adding an image to a digital certificate file			0xb428495f1fc649dd8235bfc12f9e092362a378c0cc96c4d51802a981c5bd2cbe77fe7d89ad0e1d279dbad81d31b477508ed6f0298eb589ea9fd5426d546c1944f293613344cca567031ddee2ca1d7f87c6f09ccdb263e8d160e95a1b55974d46f2851da539e06049131c2a25e5da1b	0x3c887f583330a2ae86435771d430d8f26fc2eb4	Detected	Successfully detected changes in the manipulated file

 doi:10.20944/preprints202406.0182.v1

Rotating or flipping the contents of a digital certificate file			0xb428495f1fc6 49dd8235bfc12fe455e1783edb0e 9e092362a378c0 cc96c4d51802a91146aaee29a1e6 7d89ad0e1d279 508ed6f0298eb 46c1944f293613 ee2ca1d7f87c6f 60e95a1b55974 06049131c	0x72a58ba5e263 ef4586bc8c41e2 b6c2118be62f29 1232eac4f5478c 42fb7a5616c4e4 cdab5eced0523 66e20aa5924f48 629e11b	Detected	Successfully detected changes in the manipulated file

Testing was carried out on the research object, namely the digital certificate file, where various manipulations were applied to the file to check whether changes to the file would affect the resulting signature value. If there is a change in the signature value, this indicates that manipulation of the file can be verified on the blockchain network. Table 5 is the details of testing the digital certificate file:

According to Table 5, the digital signature method using blockchain and Keccak256 hashing could find every change in the digital certificate file with 100% accuracy. Even small changes in compressed files had a big effect on their signature value.

Many studies have explored blockchain for document and digital certificate security, but some issues still need attention (Danil Muis et al., n.d.; Frikha et al., 2021; Gayathiri et al., 2020; Kelsey & Schneier, 2005; Shidqi, n.d.). One example is using blockchain without adding extra hash algorithms. This research adds to the field of blockchain-based digital certificate verification.

4. Conclusions

This research confirms that using digital signatures and the Keccak256 hash algorithm in the blockchain network can find every change in the digital certificate file with 100% accuracy. Even small changes in compressed files could be seen on the blockchain. This research shows that the designed system for verifying and validating digital certificates was successfully applied and proved effective during testing. Also, there was an improvement in the time for generating and validating signatures before and after using blockchain. The time to generate a signature without blockchain was 3.23 seconds, which reduced to 2.11 seconds with blockchain. Meanwhile, the time to validate a signature without blockchain was 2.11 seconds, which decreased to 0.22 seconds with blockchain.

Suggestions for future research include comparing different hash algorithms to assess their strengths and weaknesses in blockchain networks, and testing system performance for comparison with future studies.

Author Contributions: Conceptualization, Firman Maulana. and Irfan Darmawan; methodology, Rohmat Gunawan; software, Alam Rahmatulloh; validation, Firman Maulana., Alam Rahmatulloh. and Rohmat Gunawan; formal analysis, Irfan Darmawan; investigation, Firman Maulana; resources, Irfan Darmawan; data curation, Firman Maulana; writing—preparation of original draft, Firman Maulana; writing—reviewing and editing, Alam Rahmatulloh.; visualization, Firman Maulana; supervision, Irfan Darmawan; project administration, Rohmat Gunawan; obtaining funding, Irfan Darmawan. All authors have read and approved the published version of the manuscript.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to (specify the reason for the restriction).

Acknowledgments: In this section, you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

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

References

 doi:10.20944/preprints202406.0182.v1

1. Alshaikhli, I. F., Alahmad, M. A., & Munthir, K. (2012). Comparison and analysis study of SHA-3 finalists. *Proceedings - 2012 International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2012*, 366–371. https://doi.org/10.1109/ACSAT.2012.64
2. Bertoni, G., Daemen, J., Peeters, M., & Assche, G. Van. (2011). *Cryptographic sponge functions*.
3. Danil Muis, M., Sukarno, P., & Wardana, A. A. (n.d.). Analisis dan Implementasi Sistem Pendeteksi Ijazah dan Transkrip Palsu dengan Menggunakan IPFS dan Smart Contract Blockchain.
4. Fang, W., Chen, W., Zhang, W., Pei, J., Gao, W., & Wang, G. (2020). Digital signature scheme for information non-repudiation in blockchain: a state of the art review. *Eurasip Journal on Wireless Communications and Networking*, *2020*(1), 1–15. https://doi.org/10.1186/S13638-020-01665-W/TABLES/2
5. Foti, J. (2015). FIPS PUB 202 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions CATEGORY: COMPUTER SECURITY SUBCATEGORY: CRYPTOGRAPHY. https://doi.org/10.6028/NIST.FIPS.202
6. Frikha, T., Chaabane, F., Aouinti, N., Cheikhrouhou, O., Ben Amor, N., & Kerrouche, A. (2021). Implementation of Blockchain Consensus Algorithm on Embedded Architecture. *Security and Communication Networks*, *2021*. https://doi.org/10.1155/2021/9918697
7. Garrett, J. J. (2011). The Elements of User Experience: User-Centered Design for the Web and Beyond, Second Edition Notice of Rights Notice of Liability. www.newriders.com
8. Gayathiri, A., Jayachitra, J., & Matilda, S. (2020). Certificate validation using blockchain. *2020 7th International Conference on Smart Structures and Systems, ICSSS 2020*. https://doi.org/10.1109/ICSSS49621.2020.9201988
9. *keccak-256 Online Computing - StrErr.com*. (n.d.). Retrieved April 14, 2024, from https://www.strerr.com/en/keccak_256.html
10. Kelsey, J., & Schneier, B. (2005). Second preimages on n-bit hash functions for much less than 2n work. *Lecture Notes in Computer Science*, *3494*, 474–490. https://doi.org/10.1007/11426639_28
11. Kuznetsov, A., Oleshko, I., Tymchenko, V., Lisitsky, K., Rodinko, M., Kolhatin, A., & Karazin, V. N. (2021). Computer Network and Information Security. *Computer Network and Information Security*, *2*, 1–15. https://doi.org/10.5815/ijcnis.2021.02.01
12. Laurence, T. (2023). *Blockchain For Dummies, 3rd Edition Published*. https://books.google.com/books/about/Blockchain_For_Dummies.html?id=vbevEAAAQBAJ
13. Lorien, A., & Wellem, T. (2021). Implementasi Sistem Otentikasi Dokumen Berbasis Quick Response (QR) Code dan Digital Signature. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(4), 663–671. https://doi.org/10.29207/RESTI.V5I4.3316
14. Nadzifarin, A., & Asmunin, A. (2022). Penerapan Elliptic Curve Digital Signature Algorithm pada Tanda Tangan Digital dengan Studi Kasus Dokumen Surat – Menyurat. *Journal of Informatics and Computer Science (JINACS)*, *4*(01), 1–9. https://doi.org/10.26740/JINACS.V4N01.P1-9
15. Santoso, M. H., Girsang, N. D., Siagian, H., Wahyudi, A., & Sitorus, B. A. (2019). Perbandingan Algoritma Kriptografi Hash MD5 dan SHA-1. In *Prosiding Seminar Nasional Teknologi Informatika* (Vol. 2).
16. Saputra, I., & Nasution, S. D. (2019). Analisa Algoritma SHA-256 Untuk Mendeteksi Orisinalitas Citra Digital. *Prosiding Seminar Nasional Riset Information Science (SENARIS)*, *1*(0), 164–178. https://doi.org/10.30645/SENARIS.V1I0.20
17. Sasi, S., Subbu, S. B. V., Manoharan, P., & Abualigah, L. (2023). Design and implementation of secured file delivery protocol using enhanced elliptic curve cryptography for class I and class II transactions. *Journal of Autonomous Intelligence*, *6*(3). https://doi.org/10.32629/JAI.V6I3.740
18. Shidqi, R. M. (n.d.). Implementasi Keamanan Sistem E-Voting Pada Jaringan Berbasis Blockchain Dan Kriptografi. https://doi.org/10.13140/RG.2.2.22199.39849
19. Somsuk, K., & Thakong, M. (2020). Authentication system for e-certificate by using RSA's digital signature. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *18*(6), 2948–2955. https://doi.org/10.12928/TELKOMNIKA.V18I6.17278
20. Sowmiya, B., Poovammal, E., Ramana, K., Singh, S., & Yoon, B. (2021). Linear Elliptical Curve Digital Signature (LECDS) with Blockchain Approach for Enhanced Security on Cloud Server. *IEEE Access*, *9*, 138245–138253. https://doi.org/10.1109/ACCESS.2021.3115238
21. Stallings, W. (2013). Digital Signature Algorithms. *Cryptologia*, *37*(4), 311–327. https://doi.org/10.1080/01611194.2013.797044
22. Susilo, B., Hanyokro Kusuma, G., Hayatul Fikri, M., Saputri, R., Aulia Putri, R., Rohimah, S., Luthfi Hamzah, M., & Sultan Syarif Kasim Riau, N. (2023). Rancang Bangun Sistem Informasi Keuangan Pada Kantor Lurah Kotabaru Reteh Dengan Metode Rapid Application Develovment (RAD). *Jurnal Testing Dan Implementasi Sistem Informasi*, *1*(1), 17–28. https://www.journal.al-matani.com/index.php/jtisi/article/view/323
23. Yuniati, T., & Sidiq, M. F. (2020). Literature Review: Legalisasi Dokumen Elektronik Menggunakan Tanda Tangan Digital sebagai Alternatif Pengesahan Dokumen di Masa Pandemi. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *4*(6), 1058-1069–1058 – 1069. https://doi.org/10.29207/RESTI.V4I6.2502

 doi:10.20944/preprints202406.0182.v1

24. Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, 557–564. https://doi.org/10.1109/BIGDATACONGRESS.2017.85
25. Zou, X., & Zeng, P. (2023). A New Digital Signature Primitive and Its Application in Blockchain. *IEEE Access*, *11*, 54607–54615. https://doi.org/10.1109/ACCESS.2023.3280638