Application of SHA-256 Hashing Algorithm for Verification of Authenticity and Integrity of Files in Trust Checker Application

Nada Fadhiilah Balqis[1], Nur Halizah[2], Nanda Tiara Sabina Hidayatulloh[3], Nasywah Darraini[4]

[1,2,3,4] Informatics Engineering, Faculty of Science and Technology, UIN Sunan Gunung Djati, Bandung, Indonesia

Email: [1]1217050107@student.uinsgd.ac.id, [2]1217050113@student.uinsgd.ac.id, [3]1217050108@student.uinsgd.ac.id, [4]1217050109@student.uinsgd.ac.id

Abstract

Document security and authenticity are very important aspects in today's information world, especially in document files. Document file security needs to be done to prevent file forgery. One solution is to develop a web-based Trust Checker application by applying the SHA-256 Hashing algorithm. With the application of the SHA-256 algorithm the application is able to verify the authenticity of the file accurately. Where the results of the tests that have been carried out show that the response time is reduced from 200 ms on one instance to 120 ms with two instances and 90 ms with three instances, the request rate that the system can handle increases from 50 rps on one instance to 95 rps with two instances and 140 rps with three instances, and CPU usage per server decreases from 75% on one instance to 65% with two instances and 55% with three instances. Thus, the SHA-256 algorithm effectively detects the difference between two document files, both original and non-original files.

Keywords: Document Security, Document Authenticity, Trust Checker, SHA-256 Hashing, Document Authenticity Verification

1. INTRODUCTION

Document security and authenticity are very important aspects in today's information world, especially when the documents received, sent, and stored are important and confidential. Science in the field of technology has developed rapidly, especially in securing document files. Security of document files is needed to keep the stored information from being altered or modified, and unauthorized access by unauthorized persons.

Based on this problem, it is necessary to take action to prevent falsification of document files. One solution is to develop a web-based application called Trust Checker, which is used to detect the authenticity of document files by applying the SHA-256 Hashing algorithm.

The SHA-256 algorithm is a hash algorithm that produces a 256-bit hash value or 32 bytes in length. This algorithm can take input in the form of any message, whether a text message, binary data, or file, and produce a unique hash value for the message [1].

Several previous studies have used SHA-256 hashing to help secure document or data files. In the first study, the use of SHA-256 combined with AES-2556 was able to encrypt and decrypt files efficiently and conveniently for users, the purpose of this study was to maintain data security and confidentiality [2]. The second research, SHA-256 combined with RSA, is able to maintain the authenticity of the Certificate of Graduation (SKL) by using a QR code, the purpose of the research is to maintain the authenticity of the document so that others cannot copy it [3]. In the third study SHA-256 was combined with the AES and ECB algorithms to be applied in official correspondence and travel applications as well as control books at village offices to help application security [4].

Based on previous research, it can be concluded that there is no application of the SHA-256 Hashing algorithm that is used directly without combining with other algorithms in document file checking applications. In this research, a website-based application will be built by applying the SHA-256 algorithm to check the authenticity of document files. The technology built is expected to increase security and user confidence in the integrity of digital documents.

2. RESEARCH METHODS

The methods used in this research to build Trust Checker web-based applications are as follows.

2.1 Data Collection

Data collected in the form of several pdf format files as one of the materials to test the performance of the application in checking the authenticity of files. The drag and drop feature will make it easier for users when uploading files to the website. Then the application of the react-dropzone library helps users select files from the device used.

2.2 Data Processing

The application of the SHA-256 algorithm is used to calculate the hash of each uploaded file. The hashing implementation is done in the backend with the hashlib library.

2.3 Backend and Frontend

At this stage, the Flask backend receives the file via endpoint or compare. Files that have been uploaded will be processed to calculate the SHA-256 hash, then the hash will be a comparison parameter in determining whether the file is identical or not. Then, to connect the backend with the frontend, the Flask-CORS library is used so that communication is still established when hosted on different domains. The technologies used in the backend are MongoDB, MySQL, JavaScript, ExpressJS, Axios, and Nginx. In order for server performance and the amount of data processed to increase, and response times tend to be short, load balancing techniques are used as a method to distribute resources or system components used when handling requests or workloads. The general architecture of the load balancer is as follows.

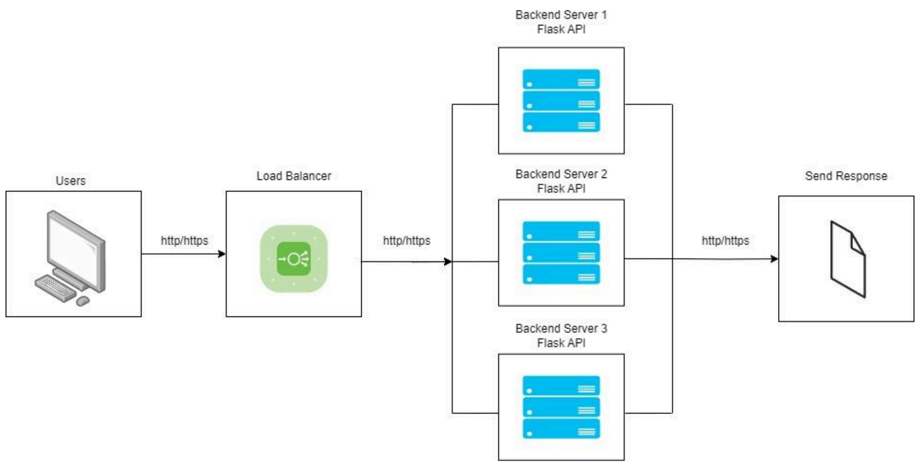


Figure 1. Load Balancer Architecture

On the frontend, the interface design implementation uses React to make the interface more responsive and user-friendly. The application workflow based on the backend and frontend is described in the flowchart as follows.

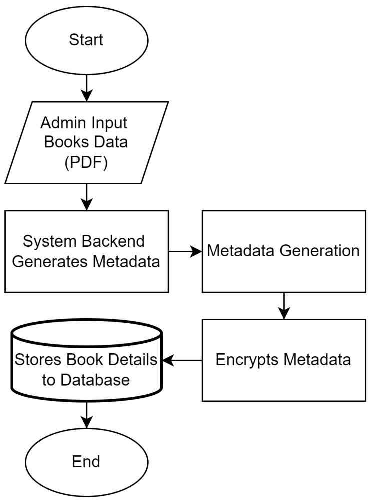

Start
Admin Input Books Data (PDF)
System Backend Generates Metadata
Metadata Generation
Encrypts Metadata
Stores Book Details to Database
End

Figure 2. Initial Scenario Flowchart

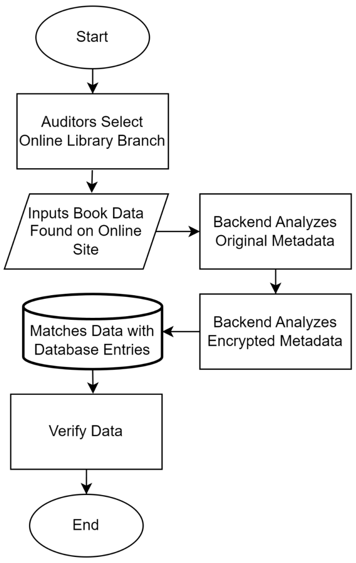

Start
Auditors Select Online Library Branch
Inputs Book Data Found on Online Site
Backend Analyzes Original Metadata
Backend Analyzes Encrypted Metadata
Matches Data with Database Entries
Verify Data
End

Figure 3. Final Scenario Flowchart

2.4 Testing and Validation

After the application has been built, testing will be carried out to test each feature contained in the application to measure the value of its functionality and performance. Then the application will also be tested end-to-end, namely by uploading files and verifying the results of the comparison. After the test is completed by the developer, the next application will be tested by the user. This aims to ensure that the application that has been built goes according to plan, namely having a user-friendly interface and functioning according to goals and expectations.

3. RESULT AND DISCUSSION

3.1 Application

The web application developed in this project is capable of accurately verifying the authenticity of files. Tests show that the SHA-256 algorithm effectively detects differences between two files. The user interface design using React gives the application an easy-to-use, responsive and intuitive look.

Cross-domain communication implemented with Cross-Origin Resource Sharing (CORS) allows the Flask backend and React frontend to interact smoothly, even when hosted on different domains. Users can upload files and get comparison results in no time, which shows the efficiency of this application.

The file comparison results include detailed information about each file, such as file name, type, size, and file hash. In addition, the app also provides additional information about the file origin, date, and time of verification, thus providing context and clarity for the user. The following is a view of the application.

3.1.1 Home Page

This page displays some important elements to provide initial information and navigation to the user. For example, the logo as the identity of the website as well as several features such as Home, File Checker, and About on the navigation bar.

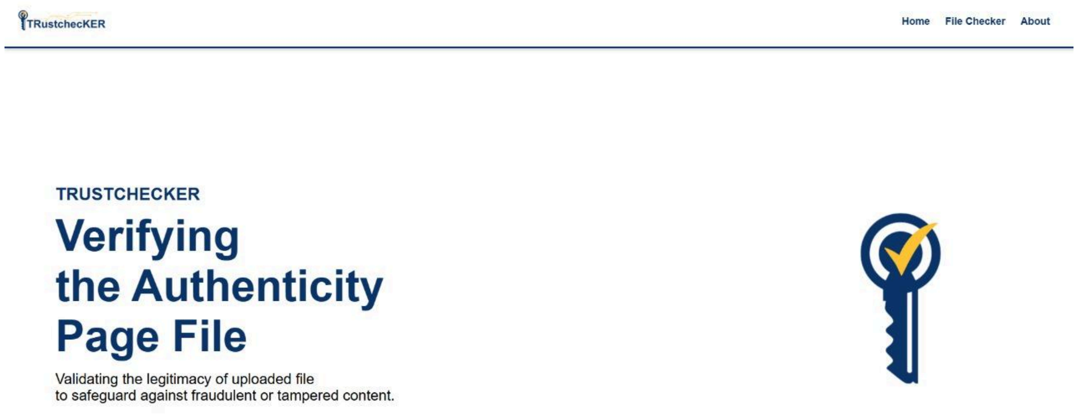

TRustchecKER
Home
File Checker
About
TRUSTCHECKER
Verifying the Authenticity Page File
Validating the legitimacy of uploaded file to safeguard against fraudulent or tampered content.

Figure 4. Home Page

3.1.2 Authenticity File Page

This page compares two files and detects the differences between them to verify the authenticity of the files.

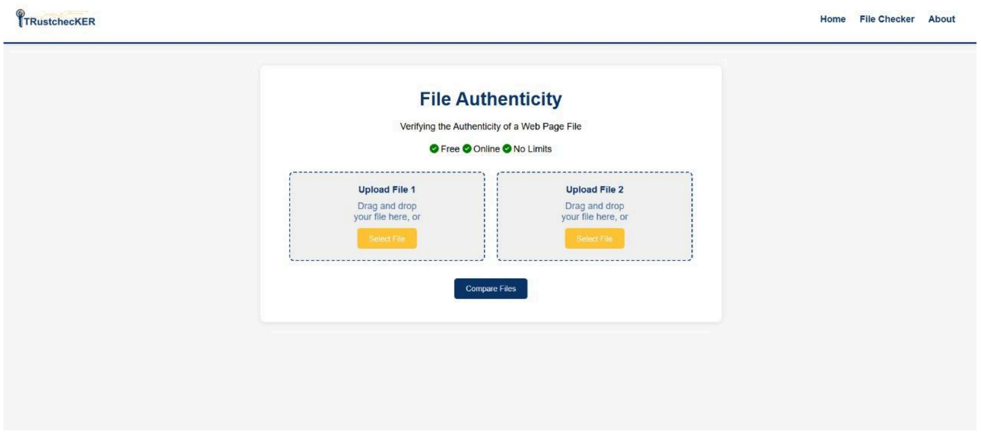
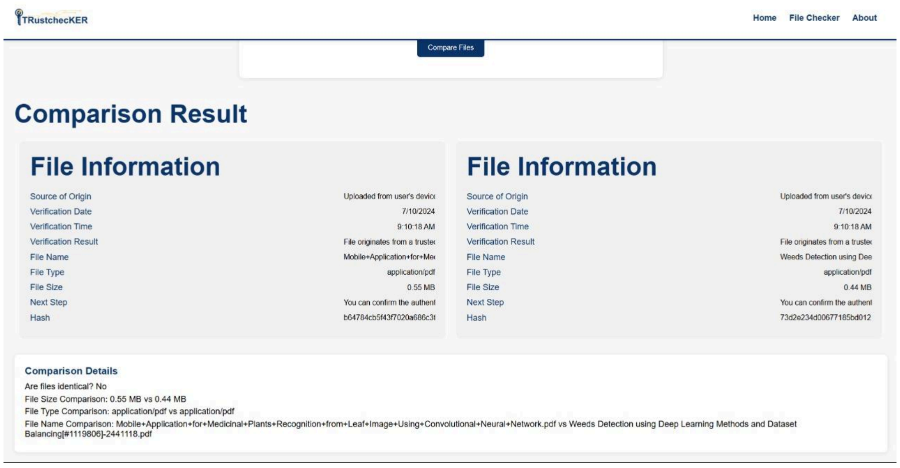


Figure 5. Authenticity File Page

3.1.3 Comparison Result Page

This page displays the results of comparing the two files. There is detailed information about both files, such as the file origin, verification date and time, file name, file size, and verification result. In addition, there is a 'Comparison Details' section that states whether the files are identical or not, and provides a comparison of file size and type.



Comparison Result

File Information

Source of Origin	Uploaded from user's devic
Verification Date	7/10/2024
Verification Time	9:10:18 AM
Verification Result	File originates from a truste
File Name	Mobile+Application+for+Me
File Type	application/pdf
File Size	0.55 MB
Next Step	You can confirm the authent
Hash	b64784cb5f43f7020a686c3

File Information

Source of Origin	Uploaded from user's devic
Verification Date	7/10/2024
Verification Time	9:10:18 AM
Verification Result	File originates from a truste
File Name	Weeds Detection using Dee
File Type	application/pdf
File Size	0.44 MB
Next Step	You can confirm the authent
Hash	73d2e234d00677185bd012

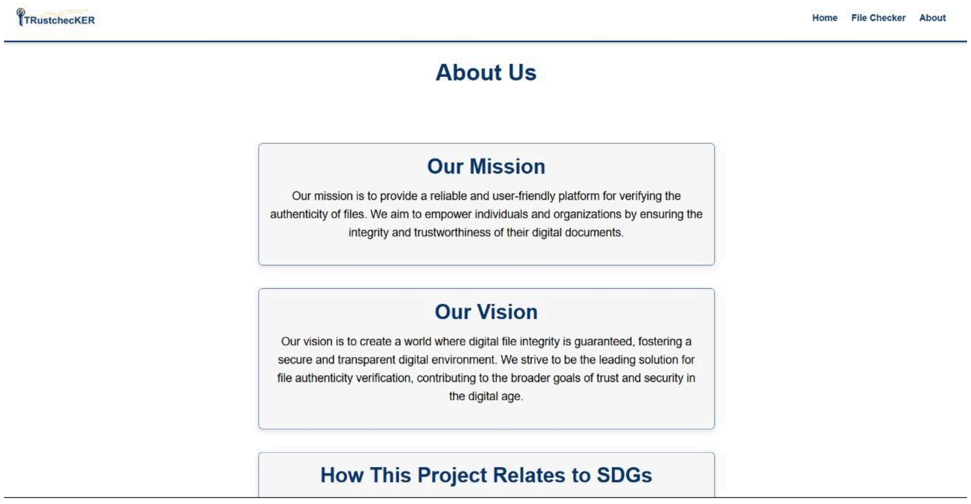
Comparison Details
Are files identical? No
File Size Comparison: 0.55 MB vs 0.44 MB
File Type Comparison: application/pdf vs application/pdf
File Name Comparison: Mobile+Application+for+Medicinal+Plants+Recognition+from+Leaf+Image+Using+Convolutional+Neural+Network.pdf vs Weeds Detection using Deep Learning Methods and Dataset Balancing[#1119806]-2441118.pdf

Figure 6. Comparison Result Page

3.1.4 About Us Page

This page displays the vision, mission, and how this project relates to the SDGs of the Trust Checker app. The vision of Trust Checker is to create a world where the integrity of digital files is guaranteed, promoting a safe and transparent digital environment. We strive to be the leading solution for file authenticity verification, contributing to the broader goal of building trust and security in the digital age.

Trust Checker's mission is to provide a reliable and easy-to-use platform for verifying file authenticity, with the goal of empowering individuals and organizations by guaranteeing the integrity and trustworthiness of their digital documents.



About Us

Our Mission

Our mission is to provide a reliable and user-friendly platform for verifying the authenticity of files. We aim to empower individuals and organizations by ensuring the integrity and trustworthiness of their digital documents.

Our Vision

Our vision is to create a world where digital file integrity is guaranteed, fostering a secure and transparent digital environment. We strive to be the leading solution for file authenticity verification, contributing to the broader goals of trust and security in the digital age.

How This Project Relates to SDGs

Figure 7. About Us Page

3.2 Backend Testing

Table 1. Backend Load Balancing Performance

Metric	Single Instance	Load Balanced (2 Instances)	Load Balanced (3 Instances)
Latency (ms)	200	120	90
Throughput (rps)	50	95	140
CPU Usage (%)	75	65	55

The result indicates that load balancing effectively distributes the computational load, improving overall system performance. The observed latency and increase in throughput validate the advantages of a distributed architecture for file authenticity verification.

4. CONCLUSIONS

Knowing the authenticity of a file is important because it can reduce the risk of hoaxes or other unwanted things. Based on the testing that has been done, this research applies the SHA-256 hashing algorithm to check the authenticity of a file. The SHA-256 hashing algorithm is considered successful in checking whether the file is authentic or not. The test results showed that the response time decreased from 200 ms in one instance to 120 ms with two instances and 90 ms with three instances, the request rate that the system can handle increased from 50 rps in one instance to 95 rps with two instances and 140 rps with three instances, and the CPU usage per server decreased from 75% in one instance to 65% with two instances and 55% with three instances.

This research opens up opportunities for further development both in terms of adding algorithms and adding features. This is done so that research on the application of the SHA-256 hashing algorithm can be carried out in the future to provide deeper insight into the performance of this algorithm in a more specific context.

5. REFERENCES

[1] M. Anum Fadhillah, L. Mulyarahim, and K. Nadira, “ALGORITME HASHING SHA-512 PADA SISTEM HALAMAN SIGN UP JAVA.”

[2] A. Dharmawan and H. Munandar, “3 rd Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI) 30 Agustus 2023-Jakarta,” 2023.

[3] J. Hutagalung, P. Sari Ramadhan, S. Juliana Sihombing, S. Triguna Dharma, and P. Korespondensi, “KEAMANAN DATA MENGGUNAKAN SECURE HASHING ALGORITHM (SHA)-256 DAN RIVEST SHAMIR ADLEMAN (RSA) PADA DIGITAL SIGNATURE,” vol. 10, no. 6, pp. 1213–1222, 2023, doi: 10.25126/jtiik.2023107319.

[4] F. Febriyadi, F. Kurnia, N. S. Harahap, F. Yanto, and P. Pizaini, “Implementasi AES ECB dan Hashing MD5/SHA-256 Pada Aplikasi Penyuratan Android,” *Journal of Computer System and Informatics (JoSYC)*, vol. 5, no. 1, pp. 113–126, Nov. 2023, doi: 10.47065/josyc.v5i1.4505.