DIGITAL FORENSICS: STRATEGIC FILE INTEGRITY VERIFICATION THROUGH CRYPTOGRAPHIC HASHING

Bekmurod G'ofurov Mansurbek o'g'li
B.Tech in Cyber Security, Sharda University Uzbekistan
https://doi.org/10.5281/zenodo.20824794

Abstract. This research presents the design, implementation, and evaluation of a File Integrity Checker based on the SHA-256 cryptographic hashing algorithm. The system enables secure verification of file integrity during transmission, storage, and forensic investigations. Unlike conventional server-based solutions, the proposed system performs all cryptographic operations locally in the user's browser using the Web Crypto API, ensuring privacy preservation and eliminating dependency on external infrastructure. Experimental evaluation demonstrates the effectiveness of the solution in detecting unauthorized modifications, malware injection, and accidental corruption.

This research presents the design, implementation, and evaluation of a File Integrity Checker based on the SHA-256 cryptographic hashing algorithm. The system enables secure verification of file integrity during transmission, storage, and forensic investigations. Unlike conventional server-based solutions, the proposed system performs all cryptographic operations locally in the user's browser using the Web Crypto API, ensuring privacy preservation and eliminating dependency on external infrastructure. Experimental evaluation demonstrates the effectiveness of the solution in detecting unauthorized modifications, malware injection, and accidental corruption.

Keywords. Digital Forensics, Cyber Security, Cryptographic Hashing, SHA-256, File Integrity Verification, Chain of Custody, Web Crypto API, React, Privacy-Preserving Systems.

Digital transformation has significantly increased the volume of digital information exchanged through networks. Ensuring the integrity of digital evidence and sensitive files is a critical requirement in cybersecurity and digital forensic investigations. Even a minor modification may compromise evidence validity. This paper proposes a practical File Integrity Checker that utilizes SHA-256 cryptographic hashing to verify whether files remain unchanged throughout transmission and storage processes.

Digital transformation has significantly increased the volume of digital information exchanged through networks. Ensuring the integrity of digital evidence and sensitive files is a critical requirement in cybersecurity and digital forensic investigations. Even a minor modification may compromise evidence validity. This paper proposes a practical File Integrity Checker that utilizes SHA-256 cryptographic hashing to verify whether files remain unchanged throughout transmission and storage processes.

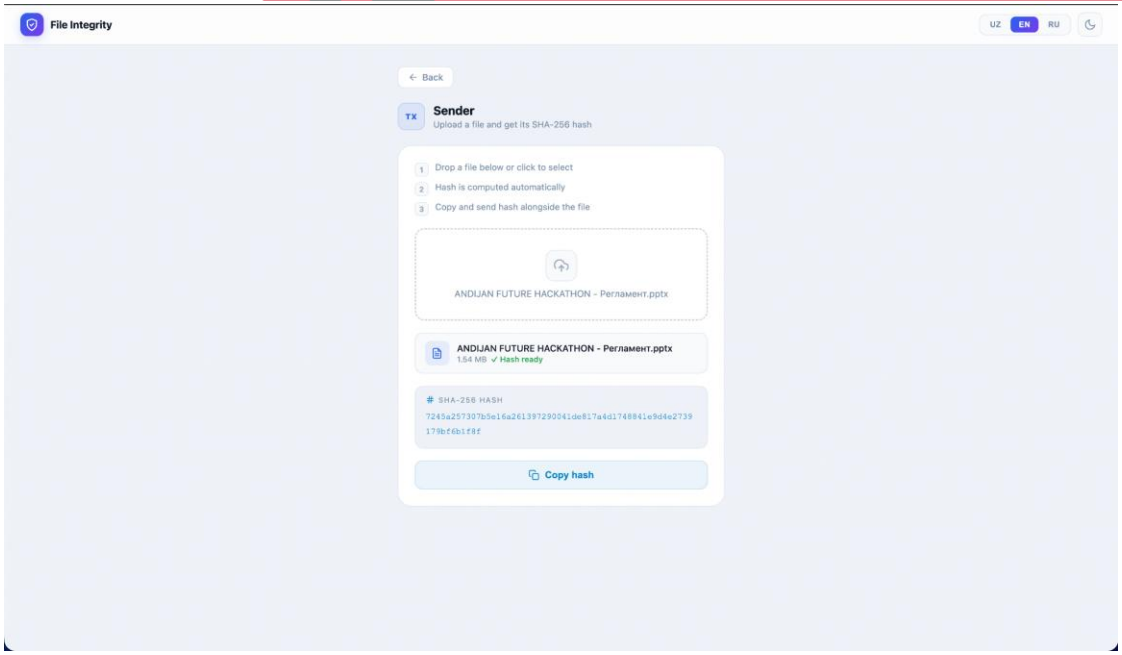

Figure 1. File Integrity Checker Interface

Cryptographic hash functions are widely used in modern security systems. NIST recommends SHA-256 for integrity verification due to its resistance against collision and preimage attacks. Previous studies demonstrate that hash-based verification remains one of the most efficient approaches for maintaining evidence integrity in digital forensics. Researchers have also emphasized the importance of preserving Chain of Custody using cryptographic methods.

Cryptographic hash functions are widely used in modern security systems. NIST recommends SHA-256 for integrity verification due to its resistance against collision and preimage attacks. Previous studies demonstrate that hash-based verification remains one of the most efficient approaches for maintaining evidence integrity in digital forensics. Researchers have also emphasized the importance of preserving Chain of Custody using cryptographic methods.

Theoretical Foundation of SHA-256. SHA-256 produces a fixed 256-bit output regardless of input size. Security relies on pre-image resistance, second pre-image resistance, collision resistance, and avalanche effect. A single-bit modification generates a dramatically different output, making unauthorized changes immediately detectable.

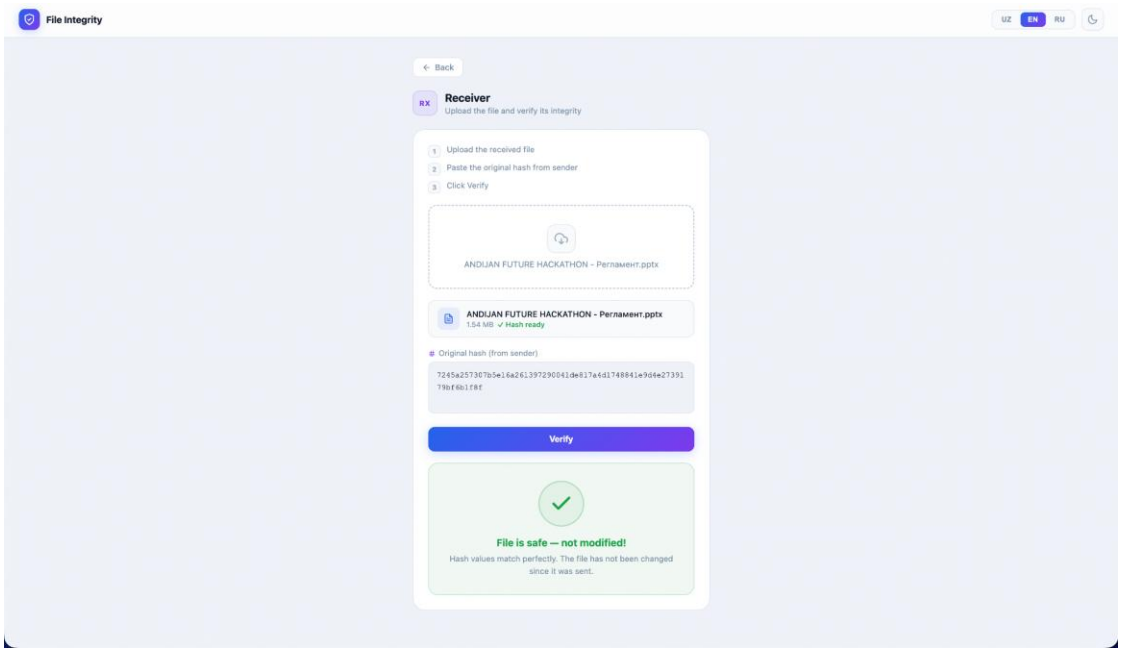

Figure 2. File Integrity Checker Interface

SHA-256 produces a fixed 256-bit output regardless of input size. Security relies on pre-image resistance, second pre-image resistance, collision resistance, and avalanche effect. A single-bit modification generates a dramatically different output, making unauthorized changes immediately detectable.

System Design and Architecture. The application was developed as a Single Page Application using React 18 and Vite. The architecture consists of Sender and Receiver modules. Sender generates a SHA-256 hash and Receiver verifies integrity by comparing original and recalculated hashes.

The application was developed as a Single Page Application using React 18 and Vite. The architecture consists of Sender and Receiver modules. Sender generates a SHA-256 hash and Receiver verifies integrity by comparing original and recalculated hashes.

Implementation Methodology. The Web Crypto API is used to compute cryptographic hashes directly within the browser. Files are processed locally and never transmitted to external servers. This privacy-first design improves trust and security. The implementation includes drag-and-drop upload, multilingual support, dark/light themes, clipboard integration, and real-time integrity verification.

The Web Crypto API is used to compute cryptographic hashes directly within the browser. Files are processed locally and never transmitted to external servers. This privacy-first design improves trust and security. The implementation includes drag-and-drop upload, multilingual support, dark/light themes, clipboard integration, and real-time integrity verification.

Workflow Analysis. The sender uploads a file, generates a hash, and transfers both file and hash to the Receiver. The Receiver recalculates the hash and compares values. Matching hashes indicate integrity preservation, while mismatches indicate tampering or corruption. The sender uploads a file, generates a hash, and transfers both file and hash to the Receiver. The Receiver recalculates the hash and compares values. Matching hashes indicate integrity preservation, while mismatches indicate tampering or corruption.

Experimental Results. Three scenarios were evaluated: unauthorized modification, malware injection, and transfer corruption. In all cases, the system correctly detected integrity violations. Results demonstrate high reliability and practical applicability in cybersecurity environments.

Three scenarios were evaluated: unauthorized modification, malware injection, and transfer corruption. In all cases, the system correctly detected integrity violations. Results demonstrate high reliability and practical applicability in cybersecurity environments.

Security Analysis. The application benefits from client-side cryptography and absence of backend infrastructure. Potential risks include social engineering and compromised endpoints, but integrity verification remains cryptographically robust. The use of native browser cryptography reduces dependency on third-party libraries. The application benefits from client-side cryptography and absence of backend infrastructure. Potential risks include social engineering and compromised endpoints, but integrity verification remains cryptographically robust. The use of native browser cryptography reduces dependency on third-party libraries.

Applications in Digital Forensics. The proposed system can support forensic investigations, evidence preservation, software distribution verification, incident response, malware analysis, and secure document exchange. Hash verification strengthens Chain of Custody requirements and legal admissibility. The proposed system can support forensic investigations, evidence preservation, software distribution verification, incident response, malware analysis, and secure document exchange. Hash verification strengthens Chain of Custody requirements and legal admissibility.

Future Work. Future development will focus on blockchain integration for immutable audit logs, automated reporting, cloud synchronization, support for multiple hashing algorithms, and enterprise-scale monitoring capabilities. Future development will focus on blockchain integration for immutable audit logs, automated reporting, cloud synchronization, support for multiple hashing algorithms, and enterprise-scale monitoring capabilities.

Conclusion. The File Integrity Checker successfully demonstrates the practical application of SHA-256 cryptographic hashing for digital forensic integrity verification. The solution combines modern web technologies with established cryptographic principles to provide an efficient, privacy-preserving, and user-friendly mechanism for ensuring file authenticity. The File Integrity Checker successfully demonstrates the practical application of SHA-256 cryptographic hashing for digital forensic integrity verification. The solution combines modern web technologies with established cryptographic principles to provide an efficient, privacy-preserving, and user-friendly mechanism for ensuring file authenticity.

Adabiyotlar, References, Литературы:

1. NIST FIPS 180-4 SHA-256 (NIST) - 2015 year
2. W3C Web Cryptography API (Ryan Sleevi and Mark Watson) - 2017
3. Cryptography and Network Security (Dr. William Stallings) - 2021
4. Cryptographic Hash-Function Basics (Phillip Rogaway and Thomas Shrimpton) - 2004
5. Source Code: https://github.com/BekmurodGofurov/File-Integrity-Checker
6. Live Application: https://file-integrity-checker.pages.dev