VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

CRYPTOGRPAHY
AAT REPORT
on

TAMPER DETECTION - SHA-384

Submitted by

Aditi Raghunandan (1BM21CS005)
Anagha M S(1BM21CS022)

Under the Guidance of
Dr. Nandhini Vineeth
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Nov-2023 to Mar-2024

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the AAT work entitled "**TAMPER DETECTION - SHA-384**" is carried out by **Aditi Raghunandan (1BM21CS005), and Anagha M S (1BM21CS022)** who are bonafide students of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraya Technological University, Belgaum during the year 2023-2024. The AAT report has been approved as it satisfies the academic requirements in respect of **Cryptography (22CS5PCCRP)** work prescribed for the said degree.

Signature of the Guide
Dr. Nandhini Vineeth
Assistant Professor
BMSCE, Bengaluru

Signature of the HOD
Dr. Jyothi S Nayak
Prof.& Head, Dept. of CSE
BMSCE, Bengaluru

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Aditi Raghunandan (1BM21CS005) and Anagha M S (1BM21CS022), students of 5th Semester, B.E, Department of Computer Science and Engineering, B. M. S. College of Engineering, Bangalore, hereby declare that, this AAT entitled "Tamper Detection-SHA-348" has been carried out by us under the guidance of Dr. Nandhini Vineeth, Assistant Professor, Department of CSE, B. M. S. College of Engineering, Bangalore during the academic semester Nov-2023-Mar-2024

We also declare that to the best of our knowledge and belief, the development reported here is not from part of any other report by any other students.

Signature

Aditi Raghunandan (1BM21CS005)

Anagha M S (1BM21CS022)

TABLE OF CONTENTS

Chapter 1:

INTRODUCTION

1.1 Literature Survey:

Windarta et al. [2023] address the increasing data integrity challenges in Internet of Things (IoT) applications by proposing two new lightweight hash functions, namely ALIT-Hash and TJUILIK-Hash. These hash functions are based on the SATURNIN block cipher and the Beetle mode of operation, specifically designed to meet the constraints of IoT systems characterized by limited computational power. A thorough security analysis was conducted , evaluating the resistance of ALIT-Hash and TJUILIK-Hash to differential and linear cryptanalysis. The proposed hash functions also underwent performance testing on both a microcontroller board and a personal computer, demonstrating competitive speeds and throughput compared to seven existing hash functions. Simulation experiments using Contiki-NG and the Cooja simulator further confirmed the favorable performance of ALIT-Hash and TJUILIK-Hash relative to other hash functions. Additionally, the cryptographic randomness tests revealed that both hash functions exhibit desirable properties. It is concluded by highlighting the cost-effectiveness and security of the proposed hash functions, making them suitable for implementation in resource-constrained IoT devices. [1]

Dhramandra Sharma and Monika Saxena [2023] focus on the critical role of cryptographic hash functions in ensuring security within blockchain technology. It provides a comprehensive overview of blockchain, its applications beyond digital currency transactions, and the necessity for protecting user privacy. Various cryptographic hash functions such as Whirlpool, MD (Message Digest), RIPEMD, SHA (Secure Hash Algorithm), and BLAKE are introduced and briefly discussed. The authors emphasize the importance of analyzing hash functions based on security criteria to assess the overall security of a blockchain system. It is concluded with comparative results, showcasing the superior performance of different cryptographic hash algorithms in terms of frequency and throughput. Notably, it identifies SHA-2 and SHA-256 as having better performance than other cryptographic hash algorithms. It highlights recent research findings related to cryptocurrency mining's environmental impact, Bitcoin market analysis, and geographical aspects of Bitcoin mining. It also discusses the potential applications of blockchain technology beyond cryptocurrency, such as in smart contracts, hyperledger, and various industries. Overall, it contributes valuable insights into the selection and evaluation of cryptographic hash functions for enhancing security in blockchain systems. [2]

Nagaeswari Bodapati et al. [2022] explores the crucial role of cryptography in ensuring information security, with a focus on digital signatures and cryptographic hash functions. As electronic transactions become increasingly prevalent, the need to protect sensitive data from malicious third parties becomes paramount.It delves into the applications of hash functions, emphasizing their significance in digital signature schemes. It discusses properties such as collision resistance, preimage resistance, and second preimage resistance as essential for secure

hash functions. It also addresses the challenge of existential forgery in digital signatures and proposes solutions involving padding and hashing to mitigate this risk. Furthermore, it examines the dominance of RSA over elliptic curve cryptography, highlighting factors such as historical precedence and simplicity of mathematics. Overall, it provides insights into the theoretical foundations and practical considerations of digital signatures and cryptographic hash functions in the context of information security. [3]

In the realm of cryptographic hash function research [2019], Kartik Rajeshwaran and Dr. Kakelli Anil Kumar introduced the Cellular Automata Based Hashing Algorithm (CABHA) as a novel approach for generating robust cryptographic hash functions. The CABHA algorithm leverages cellular automata rules, specifically Rule 30 and Rule 134, along with a custom Transformation Function to create a strong hash from an input message and a key. Unlike previous approaches, CABHA incorporates dynamic round allocation, enhancing its collision resistance and avalanche effect. The algorithm demonstrates its efficacy through thorough security analyses, showcasing its ability to produce randomized outputs for closely related messages and keys, as well as exhibiting a low probability of byte collisions. The proposed CABHA algorithm stands out for its unique combination of cellular automata principles and custom transformations, contributing to the advancement of secure hash generation methodologies in the field of cryptography. [4]

Sobti, Bagga, and Kaur [2021] introduce a novel cryptographic hash function named Cocktail, designed using the Modified ChaCha Core as its core primitive. Emphasizing simplicity, flexibility, and efficiency, Cocktail operates on the ARX (Addition, Rotation, XOR) principles, eschewing S-Boxes lookups and integer multiplications for enhanced simplicity and efficiency. The hash function is capable of working on both 32-bit and 64-bit word sizes, generating variable-sized hash outputs. Two variants, Cocktail-512 and Cocktail-1024, cater to different message digest requirements. It highlights the hash function's applications in various security contexts, including HMAC, randomized hashing, and PRF ensemble, while also demonstrating its potential for high parallelism. It provides an in-depth exploration of Cocktail's specifications, design philosophy, security aspects, and performance comparisons with SHA-3, positioning it as a viable alternative in cryptographic applications. [5]

Deena Nath Gupta and Rajendra Kumar [2021] present a comprehensive analysis of several sponge-based lightweight hash designs suitable for Internet of Things (IoT) environments. Focusing on constrained devices with limitations in battery power, storage, and transmission range, the authors evaluate six prominent hash constructions, including KECCAK, HASH-ONE, QUARK, PHOTON, SPONGENT, and GLUON. The comparative analysis encompasses parameters such as security, area requirement, digest size, throughput, power consumption, and cycle execution. The findings reveal the strengths and limitations of each design, emphasizing the need for a careful selection based on specific IoT requirements. It contributes valuable

insights to the ongoing research in lightweight cryptography, guiding researchers and practitioners in choosing optimal hash functions for secure and energy-efficient IoT applications. [6]

1.2 Problem Statement:

The current digital landscape faces a critical challenge in securing the integrity of sensitive and confidential documents during transmission. With the increasing risk of unauthorized access and tampering, existing security mechanisms in popular file formats like PDF may prove inadequate. This research aims to address the need for a robust cryptographic tamper detection approach, specifically leveraging the SHA-384 hash function, to ensure the authenticity and integrity of digital documents, particularly images . The problem encompasses the vulnerability of digital documents to tampering, the limitations of current security measures, and the potential consequences of using or distributing tampered documents. The objective is to develop and evaluate an effective solution that enhances the overall security of digital data in various sectors.

Algorithm chosen:

To tackle this issue, the research paper introduces a cryptographic tamper detection approach utilizing the **SHA-384 hash function**. Employing a 384-bit output size, SHA-384 provides a robust and computationally secure mechanism for generating hash values. This choice is driven by its resistance to collision attacks and its ability to produce unique hash representations for distinct input data.The primary goal is to establish a robust mechanism for detecting any unauthorized alterations to forensic digital data, thereby safeguarding the integrity of crucial documents transmitted through digital channels. This research underscores the urgency of implementing stringent security measures to protect against the potential exploitation of digital vulnerabilities in critical sectors.

Potential attacks on SHA-384:

Tampering Attacks:

- The primary focus is on detecting tampering attacks where an adversary modifies the content of an image to deceive or manipulate the viewer. This includes malicious alterations to the pixels, metadata, or other attributes of the image.

Cryptographic Attacks:

- The use of SHA-384 hashing indicates a concern for cryptographic attacks. Cryptographic hash functions are designed to be resistant to various attacks, including collision attacks where two different inputs produce the same hash value.

Data Integrity Attacks:

- The research application aims to safeguard against attacks that compromise the integrity of the image data. This could involve attempts to inject or modify data in a way that goes undetected by conventional means.

False Positive Injection:

- Attackers may attempt to flood a tamper detection system with false positives. This involves injecting benign or meaningless changes into the data or system to trigger false alarms. The goal is to overwhelm the detection system and create confusion.

Evasion Techniques:

- Attackers may employ evasion techniques to bypass tamper detection mechanisms. This could involve exploiting vulnerabilities in the detection system itself, using encryption to hide modifications, or carefully crafting attacks to avoid detection.

Stealthy Tampering:

- Sophisticated attackers may attempt to tamper with systems or data in a subtle and stealthy manner to avoid triggering immediate detection. This involves making minimal and carefully planned changes that are less likely to be noticed by the tamper detection system.

Side-Channel Attacks:

- Side-channel attacks involve exploiting information leaked during the tamper detection process, such as timing information or power consumption patterns. By analyzing these side-channel signals, attackers may gain insights into the workings of the tamper detection mechanism and find ways to evade it.

Tampering with Detection Sensors:

- Some tamper detection mechanisms rely on physical sensors or hardware-based mechanisms to detect tampering. Attackers may attempt to manipulate or disable these sensors, rendering the detection system ineffective.

Reverse Engineering:

- Attackers may reverse engineer the tamper detection software or hardware to understand its inner workings. Once the detection mechanism is understood, attackers can develop methods to bypass or manipulate it without triggering alarms.

Denial-of-Service (DoS) Attacks:

- Tamper detection systems may be vulnerable to denial-of-service attacks that overwhelm their resources, making them unable to perform proper monitoring and detection. This could be achieved through flooding the system with traffic or exploiting vulnerabilities in the detection software.

Physical Attacks:

- For systems that rely on physical tamper-evident seals or mechanisms, attackers may physically manipulate or replace these seals without triggering detection. This could involve carefully opening a sealed container and resealing it to appear untouched.

1.3 Motivation:

The motivation behind choosing the identified problem statement lies in the increasing reliance on digital communication and the critical need for ensuring the integrity and authenticity of digital documents. In today's technologically driven landscape, the widespread use of digital platforms for document exchange has become integral to various sectors, including government, finance, and defense.

The escalating sophistication of tampering tools poses a genuine threat to the security of digital information. Instances of unauthorized alterations to crucial documents can lead to severe consequences, ranging from financial fraud and legal disputes to compromising national security. As such, the motivation stems from the recognition that existing security measures may fall short in the face of evolving cyber threats, necessitating innovative solutions to address vulnerabilities in the digital communication infrastructure.

Moreover, the potential misuse of tampered documents could erode public trust in digital transactions and communication, hindering the seamless functioning of institutions and organizations. The motivation for this research lies in the overarching goal of enhancing the reliability and security of digital exchanges by proposing a cryptographic tamper detection approach. By addressing this motivation, the research aims to contribute to the development of robust mechanisms that can be widely adopted across sectors to safeguard the integrity of digital documents and fortify the overall cybersecurity landscape.

1.4 Various Aspects of the Algorithm:

The algorithm chosen for the proposed cryptographic tamper detection approach encompasses several key aspects designed to ensure effectiveness, efficiency, and adaptability. These aspects include:

1) Hash Function Selection:

The choice of a robust cryptographic hash function is crucial. The selected algorithm likely employs a well-established hash function known for its collision resistance and cryptographic strength. Popular choices include SHA-256 or SHA-3.

2) Tamper Detection Mechanism:

The algorithm incorporates a tamper detection mechanism that involves generating and verifying digital signatures or hashes. This mechanism allows the system to detect any unauthorized alterations to the digital document.

3) Key Management:

Efficient key management is essential for the security of the cryptographic system. The algorithm likely outlines a method for key generation, distribution, and secure storage to prevent unauthorized access.

4) Computational Efficiency:

The efficiency of the algorithm is considered, taking into account the computational resources required for both generating and verifying digital signatures or hashes. A balance between security and computational efficiency is crucial for real-world applicability.

5) Scalability:

The algorithm is designed to be scalable, accommodating various document sizes and types. It should be applicable to a wide range of digital documents, from small text files to large multimedia files.

6) Resistance to Attacks:

The algorithm considers potential attacks on cryptographic systems, such as collision attacks, and incorporates measures to resist these attacks. It aims to provide a high level of security against various tampering attempts.

7) Interoperability:

Consideration is given to interoperability, ensuring that the algorithm can be integrated into existing digital communication and document management systems without significant modifications. This facilitates widespread adoption across different platforms and applications.

8) Documentation and Standardization:

The algorithm is likely documented thoroughly, providing clear guidelines for implementation, usage, and integration. Standardization efforts may be considered to enhance the algorithm's credibility and promote its adoption in the industry.

9) Adaptability to Emerging Threats:

The algorithm takes into account the dynamic nature of cybersecurity threats. It may include provisions for updates and modifications to adapt to emerging threats and technological advancements, ensuring its relevance over time.

By addressing these various aspects, the chosen algorithm aims to provide a comprehensive and reliable solution for cryptographic tamper detection in digital documents.

Chapter 2:

METHODOLOGY

2.1 Procedure:

The code implements an Image Tampering Detection system using a Flask web application. The system uses a hash function based on the SHA-384 algorithm to generate hashes for different components of the image and checks for tampering based on these hashes. Here are the various steps and explanations for the code:

1. Hashing Algorithm (SHA-384):

The SHA-384 algorithm is a cryptographic hash function that takes an input message and produces a fixed-size output hash (384 bits or 48 bytes). The provided code includes functions to process blocks and calculate the SHA-384 hash.

- rotate_right(val, n): A helper function to perform bitwise rotation to the right.
- sha384_padding(message_len): Generates the padding needed for the input message to be a multiple of 1024 bits (128 bytes) long.

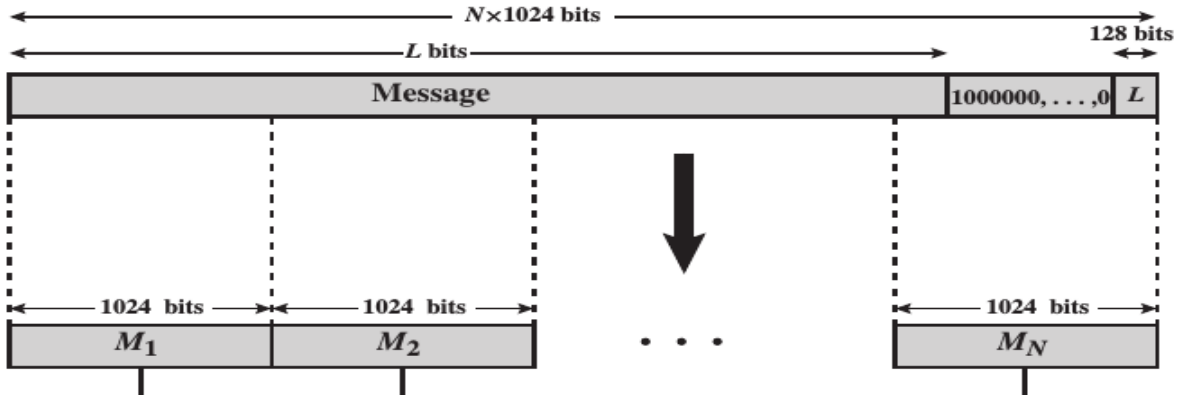

Fig 2.1.1 - Message padding illustration

- sha384_process_block(block, h): Processes a 1024-bit block of the input message and updates the hash values.
    - Divide the padded message into 1024-bit blocks.
    - The block is divided into 16 words initially.
    - Before processing the message blocks, the eight 64-bit words A, B, C, D, E, F, G, H are initialized with specific constant values derived from the fractional parts of

the square roots of prime numbers. This initialization happens once at the beginning of the entire hashing process.

- During the processing rounds, these 16 words are expanded into 80 words through a specific schedule, which involves bitwise and logical operations.
- The first 16 words of the message schedule, denoted as Wt with t varying from 0 to 15 and are initially filled with 64bits each.
- For t=16 to 79 the remaining 64 words are computed using the following formula:

$$Wt = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$$

Fig 2.1.2 - Logical functions applied to words

- $\sigma_0(x) = \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus (x \gg 7)$
- $\sigma_1(x) = \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus (x \gg 6)$

Here, $\mathrm{ROTR}^n(x)$ represents a right circular rotation of x by n positions, and $\gg$ represents a right shift.

Fig 2.1.3 - Bitwise logical functions

The resulting message schedule *Wt* is then used in the processing rounds of the SHA-384 algorithm. The expanded words are combined with the current state (hash values) to compute new hash values during each iteration.

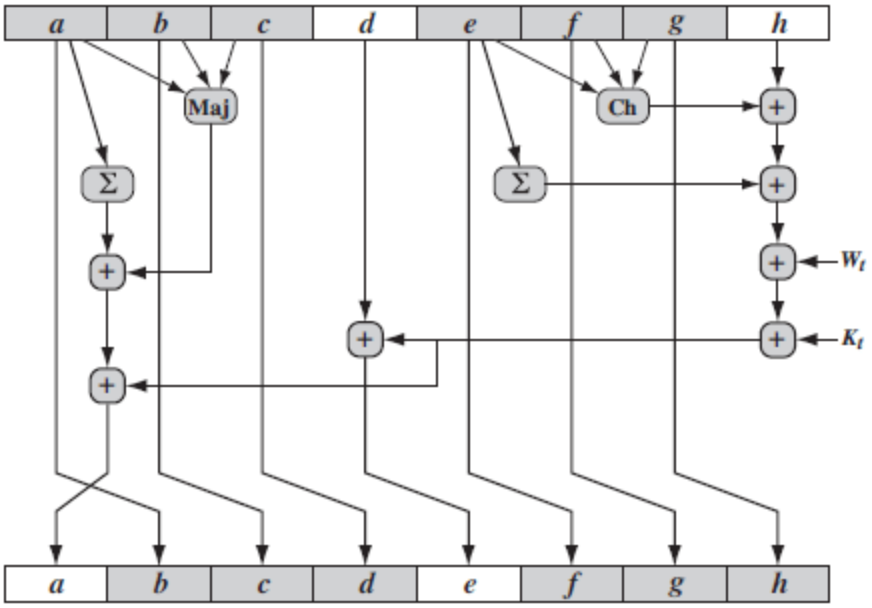

Fig 2.1.4 - Diagrammatic representation of single round

- $Ch(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G)$
- $Maj(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$
- $\Sigma_0(A) = (\mathrm{ROTR}^{28}(A) \oplus \mathrm{ROTR}^{34}(A) \oplus \mathrm{ROTR}^{39}(A))$
- $\Sigma_1(E) = (\mathrm{ROTR}^{14}(E) \oplus \mathrm{ROTR}^{18}(E) \oplus \mathrm{ROTR}^{41}(E))$

Here, $\wedge$ is bitwise AND, $\oplus$ is bitwise XOR, $\neg$ is bitwise NOT, and ROTR^n is a right circular rotation by n positions.

Fig 2.1.5 - Core operation in each round

- sha384_hash(data): Takes an input message, pads it, and processes it in blocks to produce the final SHA-384 hash.
    - After processing all blocks, the final hash value is obtained by concatenating the hash values
    - Perform 80 iterations of the SHA-384 processing function on the first 1024-bit data block . The resulting 384-bit output initializes A, B, C, D, E, F, G, and H for the processing function of the next data x After all N data blocks have been processed, the final output forms the 384-bit message digest

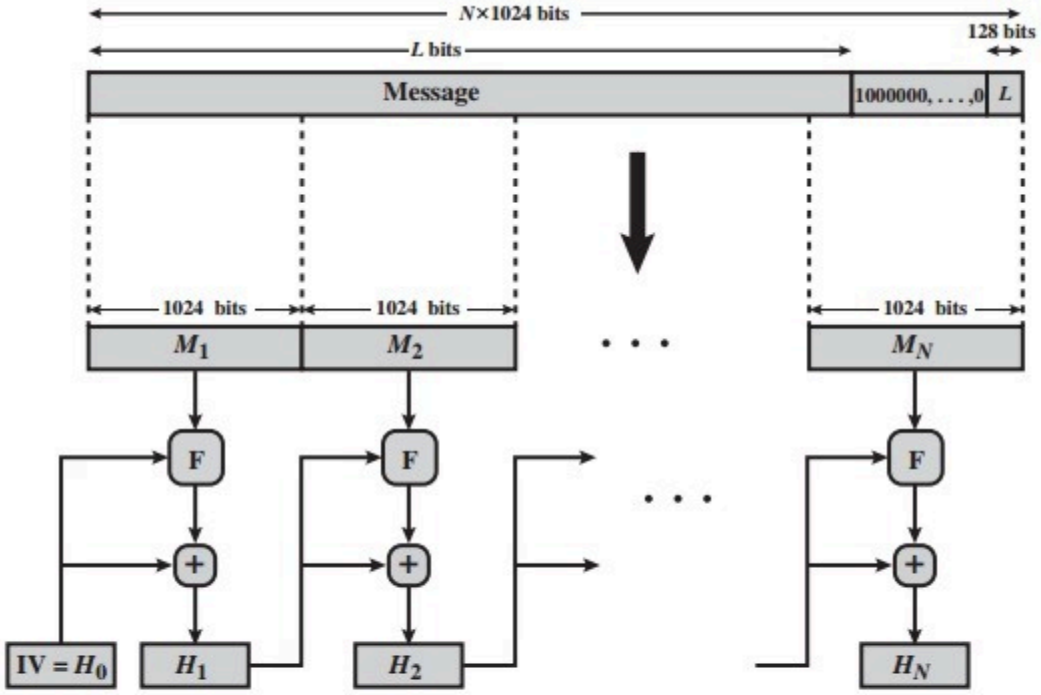

Fig 2.1.6 - Overall Block Diagram

2. Image Hashing:

- The hash_image function takes the image path, opens the image using PIL (Python Imaging Library), converts it to bytes, and then computes the SHA-384 hash using the previously defined functions.

3. Checking Tampering:

- The check_tampering function checks for tampering in the provided image. It calculates the SHA-384 hash for the entire image and for each color channel (Red, Green, Blue). It then concatenates these hashes and generates a connected hash.

4. Flask Web Application:

- app.route('/'): Renders the initial page with URLs for original and tampered images.
- app.route('/check_tampering', methods=['POST']): Handles the form submission, saves the uploaded images, and checks for tampering. Renders the result page.

5. HTML Templates:

- index.html provides a form for uploading original and tampered images.
- result.html displays the result of the tampering check.

6. Execution:

- When the script is executed, the Flask development server starts, and the web application becomes accessible at http://127.0.0.1:5000/.
- Users can visit the main page, upload original and tampered images, and submit the form to check for tampering.

2.2 Flow Chart:

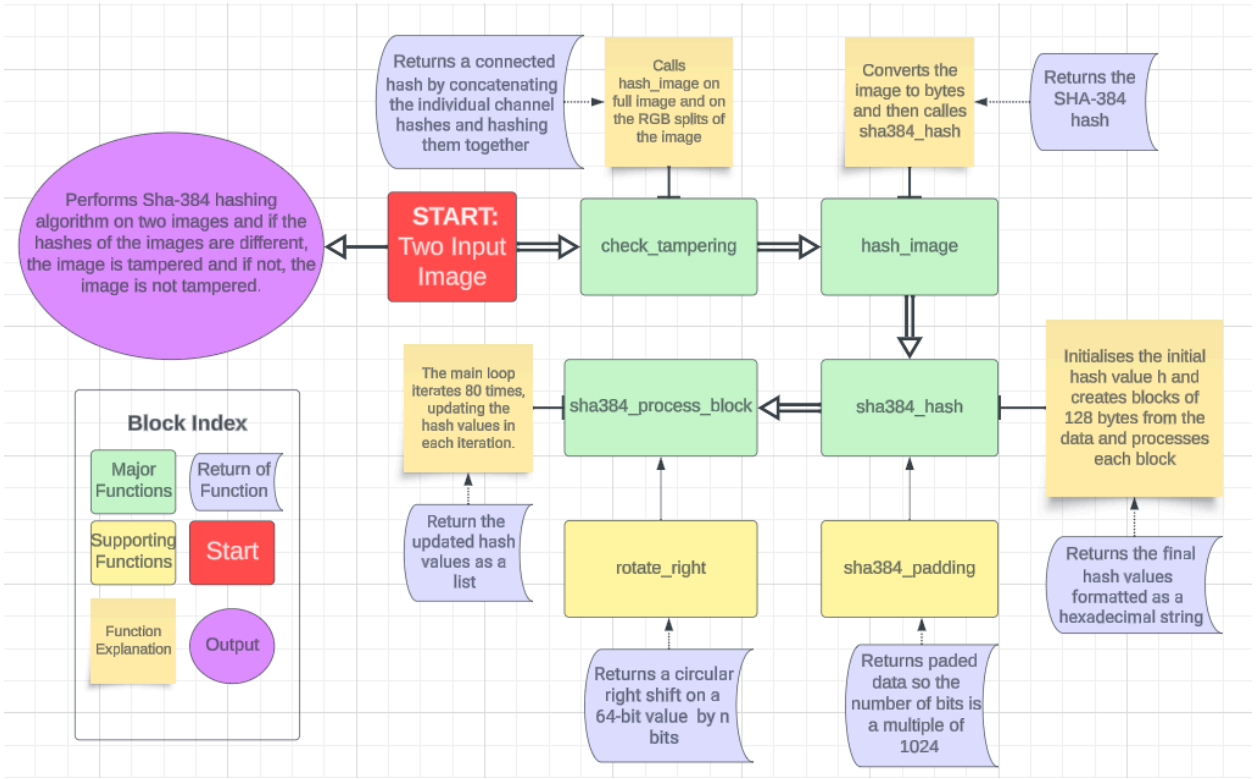

Fig 2.2.1 - Flow of the entire code

Chapter 3:

Results and Discussion

3.1 Results:

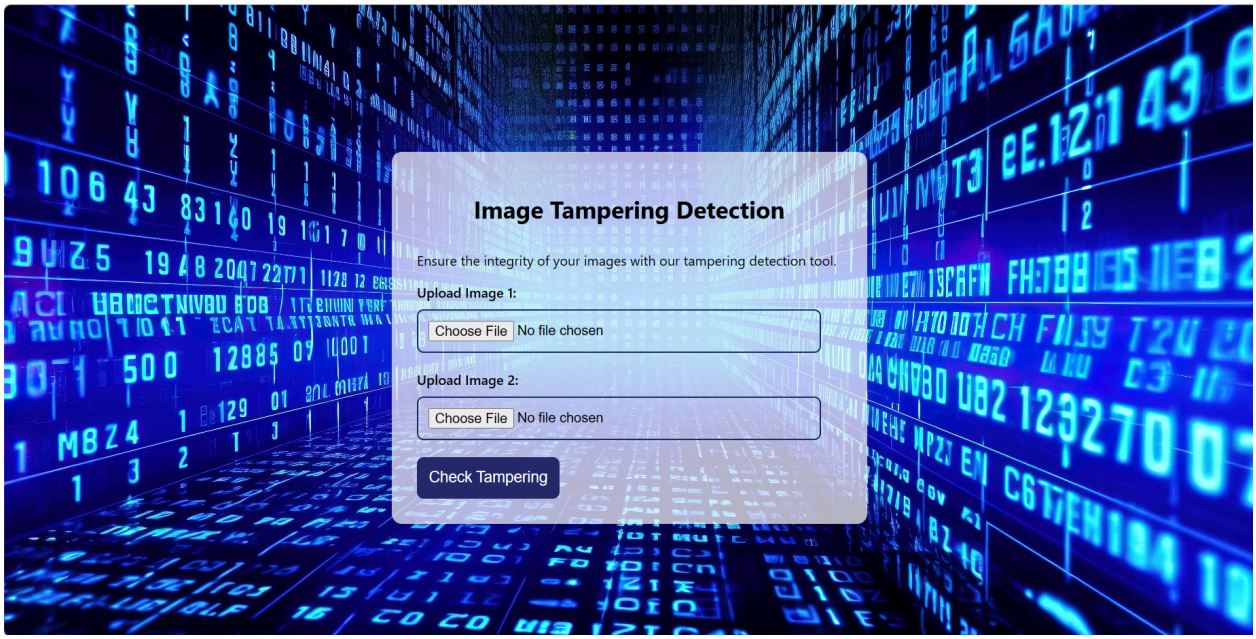

Fig 3.1.1- Front page of the interface. We can select the images to check for tampering here.

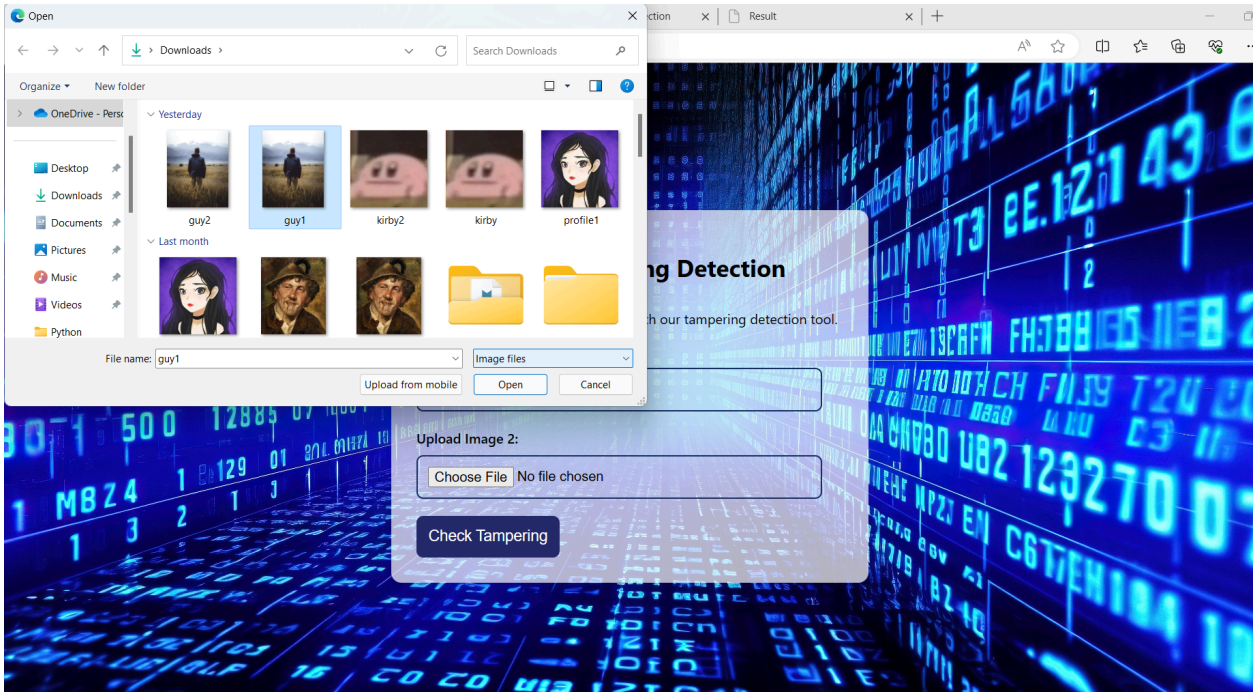

Fig 3.1.2- Choosing two files from file explorer.

Case 1:

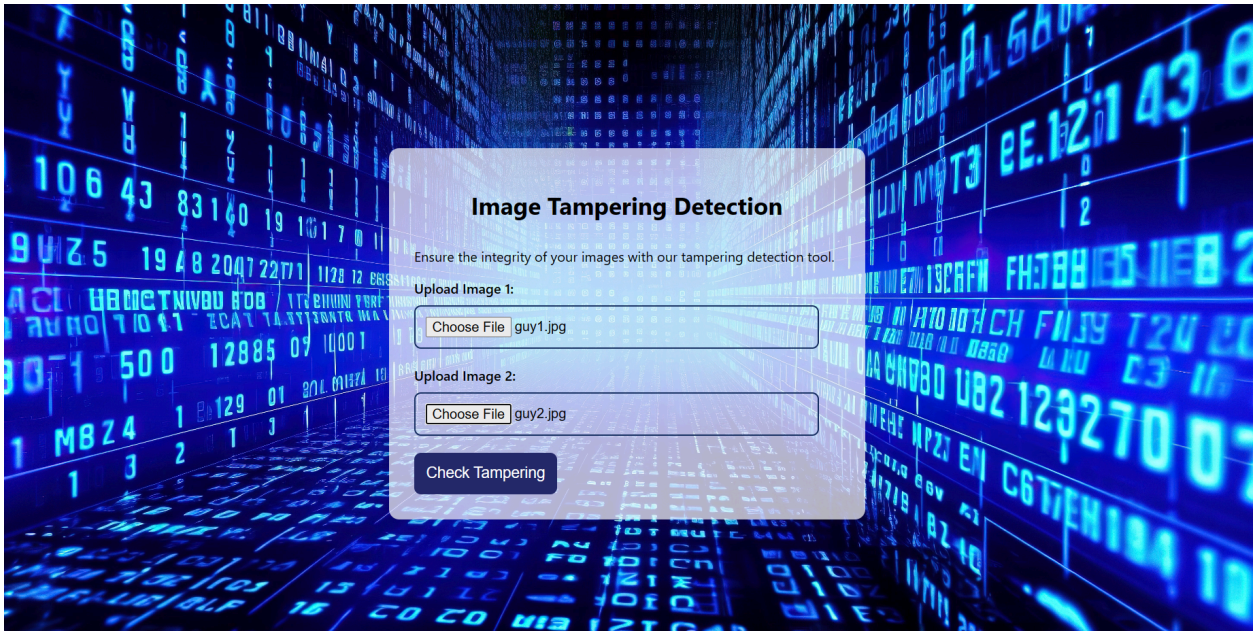

Fig 3.1.3- The names of the chosen files displayed.

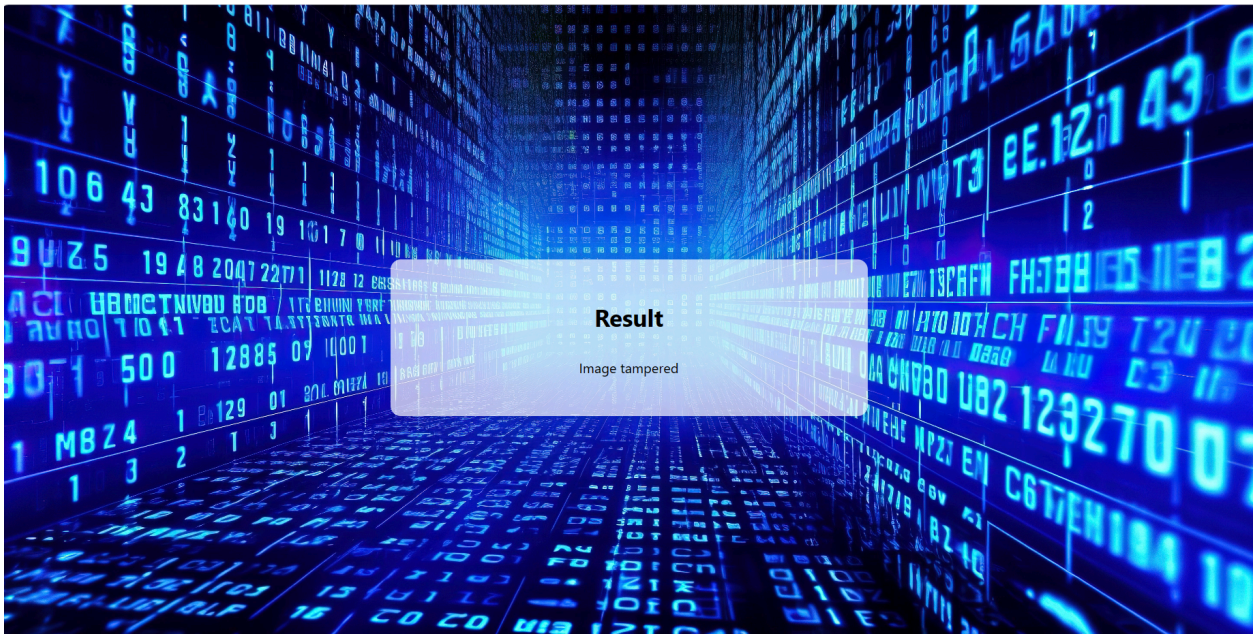

Fig 3.1.4- The result page displays the output. The image is tampered.

Case 2:

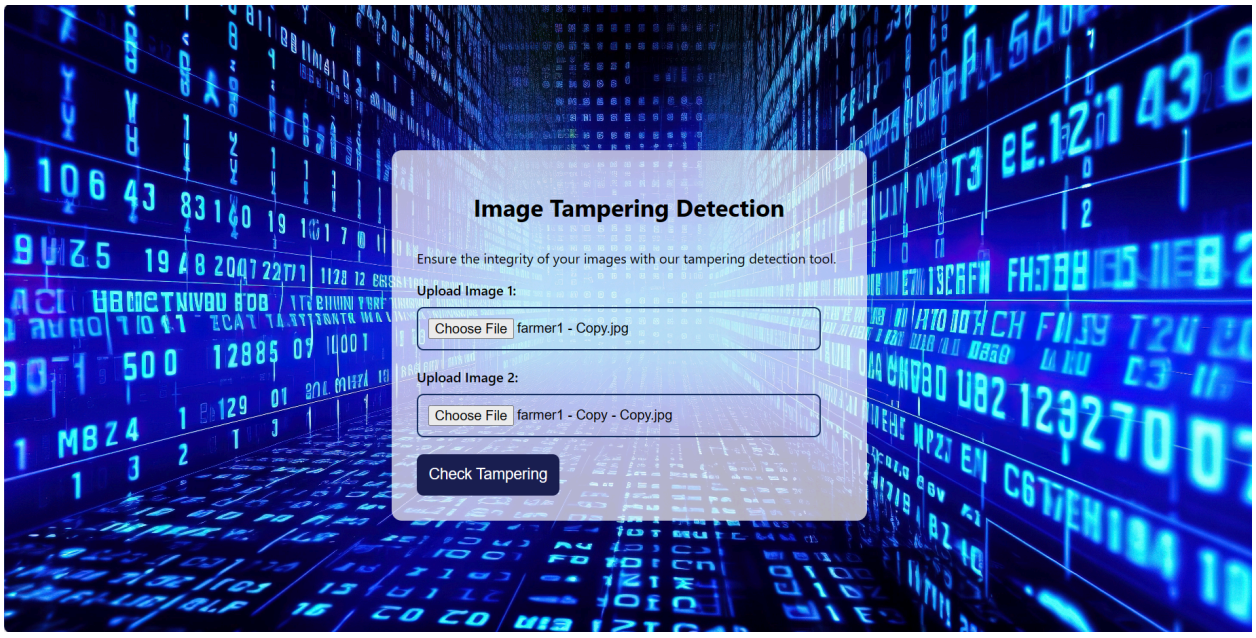

Fig 3.1.5- Choosing two other images.

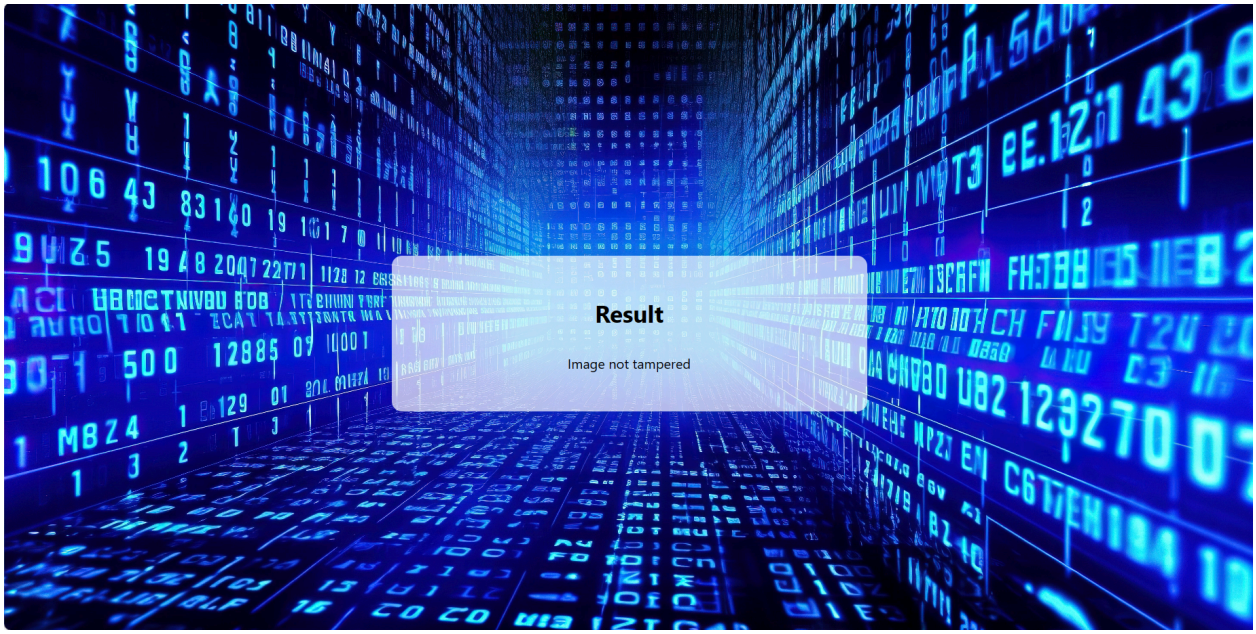

Fig 3.1.6- The result page displays the output. The image is not tampered.

3.2 Discussion:

SHA-384 Hashing:

- The code includes a complete implementation of the SHA-384 hashing algorithm.
- Images are hashed using SHA-384 for the entire image and individual color channels (Red, Green, Blue).

Image Tampering Detection:

- The tampering detection is based on the comparison of SHA-384 hashes.
- The check_tampering function checks whether the original and tampered images are identical by comparing their SHA-384 hashes.

Connected Hashing:

- The code concatenates the hashes of the entire image and its color channels to create a connected hash.
- The connected hash is then used to determine whether the images have been tampered with.

Flask Web Application:

- The code implements a Flask web application for users to upload original and tampered images.
- The web application provides a user-friendly interface for conducting tampering checks.

Web Interface Results:

- The result of the tampering check is presented to the user on a web page ('result.html').
- The user receives a clear indication of whether the images are tampered or not.

Results Visualization:

- The document includes visual representations of sample images and their corresponding RGB channels.
- The SHA-384 hash values for each channel are provided in hexadecimal, base64, and other formats.

Chapter 4:

CONCLUSION AND FUTURE WORK

4.1 Conclusion:

Tamper Detection Strength:

- The use of SHA-384 hash function is presented as an effective method for detecting tampering in digital documents.
- Changes in pixel values in images can be easily detected through alterations in hash values, enhancing document security.
- SHA-384 is a strong cryptographic hash function, providing a high level of security.
- The use of SHA-384 enhances the robustness of the tampering detection system.

Connected Hash Values:

- The strength of the connected hash values is highlighted, suggesting that the security of the system is dependent on the length of the chain of hashes.
- Encouragement is given for the use of multiple linked connected hashes in a centralized or distributed database for increased security.

Practical Application:

- The proposed approach is positioned as a practical method for experts to detect document tampering.
- The impact of changes in pixel values on hash values adds a layer of security to document integrity.

Encouragement for Database Security:

- The conclusion emphasizes the importance of maintaining multiple linked connected hashes in a database for higher security.
- This implies that a centralized or distributed database structure with a chain of linked hashes can enhance the overall security of digital documents.

4.2 Future Work:

To enhance the effectiveness and versatility of the image tampering detection system, several potential improvements and future enhancements can be considered. First and foremost, implementing comprehensive code documentation and strengthening error handling mechanisms are crucial steps to ensure code readability, maintainability, and robustness. This includes detailed comments and documentation to clarify the purpose of each function and segment of the code. Robust error handling will help manage potential exceptions during image processing and web application interactions, contributing to a more reliable and secure system.

In terms of future enhancements, exploring the integration of machine learning techniques for tampering detection can significantly boost accuracy, especially in scenarios involving

sophisticated tampering methods. Additionally, the incorporation of real-time monitoring features would make the system suitable for situations where images are subject to ongoing changes. Offering customization options, such as allowing users to choose different hash algorithms, adjust sensitivity levels, or even extending support for a broader range of image formats, would provide users with a more tailored and flexible tampering detection experience. Furthermore, user authentication mechanisms can be introduced for scenarios involving sensitive or private image data, ensuring secure access. These future enhancements collectively aim to elevate the system's capabilities, making it more adaptive, feature-rich, and user-friendly.

Chapter 5

REFERENCES

[1] S. Windarta *et al*., "Two New Lightweight Cryptographic Hash Functions Based on Saturnin and Beetle for the Internet of Things," in *IEEE Access*, vol. 11, 2023

[2] D. Sharma and M. Saxena, "Different Cryptographic Hash Functions for Security in the Blockchain," *2023 International Conference on Data Science and Network Security (ICDSNS)*, Tiptur, India, 2023

[3] N. Bodapati, N. Pooja, E. A. Varshini and R. N. S. Jyothi, "Observations on the Theory of Digital Signatures and Cryptographic Hash Functions," *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, 2022

[4] K. Rajeshwaran and K. Anil Kumar, "Cellular Automata Based Hashing Algorithm (CABHA) for Strong Cryptographic Hash Function," *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, 2019

[5] R. Sobti, A. Bagga and P. Kaur, "Cocktail - An ARX based Cryptographic Hash Function Designed using Modified ChaCha Core," *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, 2021

[6] D. N. Gupta and R. Kumar, "Sponge based Lightweight Cryptographic Hash Functions for IoT Applications," *2021 International Conference on Intelligent Technologies (CONIT)*, Hubli, India, 2021