Journal of Electronics and Informatics (ISSN: 2582-3825)
www.irojournals.com/iroei/

A Comparative Study on Hashing Algorithms for Data Integrity and Efficiency

Atshaya R.[1], Bhavatharni J.[2], Darshana S.B.[3], Ismankhan Y.M.[4]

Computer Science and Engineering, Sri Ramakrishna Engineering College, Coimbatore, India

E-mail: [1]atshaya.2201028@srec.ac.in, [2]bhavatharni.2201030@srec.ac.in, [3]darshana.2201033@srec.ac.in, [4]ismankhan.ym@srec.ac.in

Abstract

In recent years, the widespread availability of image editing tools has led to the proliferation of phony and manipulated photos on the Internet and social media. Various techniques have been developed to detect image forgery and identify altered or fabricated regions, with a growing emphasis on deep learning (DL) methods. This study explores recent advances in DL-based forgery detection algorithms, focusing on the detection of copy-move and splicing attacks two of the most common image tampering techniques. Additionally, the challenges posed by DeepFake-generated content, which often mimics splicing manipulation, are discussed. The study also compares hashing algorithms (SHA-256, CRC32, Random Projection Hashing, and Count-Min Sketch) for use in data integrity, similarity searches, and frequency estimation. Finally, recommendations for selecting suitable algorithms and hybrid approaches are provided to enhance image authentication and large-scale data analysis.

Keywords: Image forgery detection, deep learning, copy-move attack, splicing attack, Deep Fake detection, data integrity, hashing algorithms, SHA-256, CRC32, Random Projection Hashing, Count-Min Sketch, similarity detection, streaming data analytics, probabilistic data structures

1. Introduction

Image comparison is a critical operation in applications such as digital forensics, content authentication, and multimedia analysis. Verification of authenticity using image similarity and forgery detection is essential. This article proposes a system developed using the

DOI: https://doi.org/10.36548/jei.2025.2.002
Received: 03.04.2025, received in revised form: 27.04.2025, accepted: 11.05.2025, published: 20.05.2025

Django framework based on hashing and pixel-wise comparison approaches to determine image similarity and detect tampering. The system is modular, with components designed for efficient processing, extensive analysis, and simple result presentation. Software such as Photoshop, GIMP, and CorelDraw makes it difficult to identify a tampered image from an original image. Most of the conventional image forgery detection methods are based on manually extracted features. The disadvantage of these conventional methods is that most are capable of detecting different types of tampering by searching for specific features in the image. Nowadays, image forgery detection is performed using deep learning-based techniques. Thee capability of these techniques to extract complex information from images has led to claims of higher accuracy compared to conventional methods. In this work, we present a comprehensive study and results of deep learning-based techniques for image forgery detection, along with information about publicly available image forgery datasets.

2. Related work

Image forgery detection has progressed from conventional statistical methods to advanced deep learning architectures, overcoming issues presented by tampered digital content. Initial attempts were based on detecting lighting inconsistencies [3] and chromatic aberration [4], whereas signal-based methods like moment features with the Hilbert-Huang Transform [5] and higher-order statistics [6] were used to unveil splicing. Natural image models were utilized to detect spliced areas [7]. With the advent of deep learning, convolutional neural networks (CNNs) took center stage, with models identifying copy-move forgeries [2], and Busternet exhibiting source/target localization ability [8]. More recent research by Ali et al. [1] introduced the use of recompression-based preprocessing to improve CNN performance. Surveys by Zanardelli et al. [11], Singh and Kumar [12], Deb et al. [13], and Pham and Park [15] thoroughly examine these trends, pointing out important challenges like generalization across datasets, post-processing robustness, and explainability. Real-world implementations such as Python- and OpenCV-based detection tools [9][10] and MD5 checksum verification [9]. The use of deep learning for inpainting forgery [14] and the investigation of end-to-end systems for hybrid manipulation detection highlight the current direction of research. Together, the area is moving toward stronger, data-driven systems capable of identifying more sophisticated forgeries in real-world applications.

Table 1. Comparison of Existing Methods

Sr. No.	Author	Method Used	Forgery Type	Model	Accuracy
1.	Ali S.S.[1]	Recompression and CNN	Copy move and Image splicing	CNN	67.71 61.31 61.83 62.23
2.	Wu Y, Abd-Almageed W, Natarajan P	BusterNet (pixel based)	Copy-move	VGG16	76% - 80.49%
3.	Shi, Y.Q., Chen, C., Chen, W [7]	Moments of characteristic functions of wavelet subbands	Image splicing	SVM with RBF kernel	61.87%
4.	Fu et al. [5]	HHT and Wavelet Decomposition	Image splicing	SVM	80.15%
5.	Ng, T., Chang, S., Sun, Q [6]	Bicoherence features	Image splicing	SVM	62-70%

Table 1 presents the accuracy of the method used, the forgery type, and the model employed in the existing studies of different authors.

3. Proposed Work

The Input Module is the basis of the image comparison pipeline by ensuring that the input images supplied are in the correct format and conform to specific requirements prior to being further processed. The module is tasked with accepting user-uploaded image files via a web interface based on Django. The system will check the input against several parameters, including file type, size, resolution, and color depth, in an effort to block incompatible or corrupted images from being processed.

Validation of files is performed through Django's built-in validators and custom logic to verify the MIME type and file extensions (e.g., JPEG, PNG, BMP). In addition, the module applies size restrictions to ensure efficiency and prevent performance bottlenecks during comparison. Invalid files trigger informative error messages, guiding users to acceptable input formats. This rigorous validation pipeline guarantees the integrity of the subsequent analysis while offering a user-friendly interface. The Input Module is the central component of the image comparison pipeline, responsible for ensuring that only valid and suitable image files are passed to further processing. It accepts user-uploaded images through a Django-based web interface with support for multiple formats, including JPEG, PNG, BMP, and TIFF.

Rigorous validation criteria are utilized to guarantee the integrity and compatibility of the images. These include MIME type checks, file extension checks, resolution consistency (e.g., a minimum of 512x512 pixels), and size restrictions to prevent performance bottlenecks with extremely large files. In addition, the module ensures color depth consistency (e.g., 8-bit or 16-bit per channel) to guarantee accuracy during comparison. Invalid files trigger descriptive error messages, such as the reporting of unsupported formats or large files, thus guiding users to correct errors. This rigorous validation pipeline ensures that all images processed are of sufficient quality for meaningful comparison, providing a sound foundation for follow-up analytical modules.

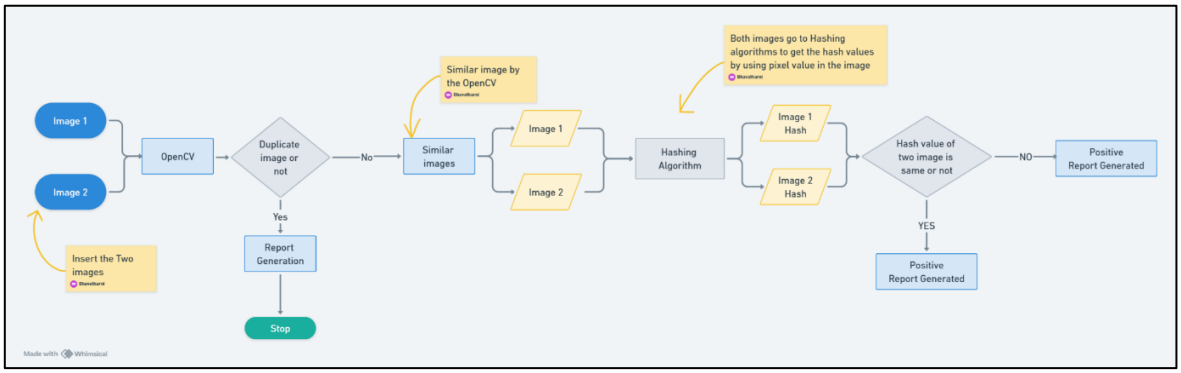

Figure 1. Work Flow

Fig.1 shows the workflow of the image forgery detection process. Two images are given as input, which are processed by the OpenCV library. In the pixel comparison, if the pixels vary, then hashing algorithms are applied. If the hash values differ, then the report is generated.

System Workflow

Step 1: Image Input

- The process begins with the uploading of two images that are to be compared for similarity or forgeries.

Step 2: Similarity Check with OpenCV

- OpenCV is utilized to conduct an initial comparison. The process compares the visual similarity between the two images, representing the probability of duplication or manipulation.

Step 3: Duplicate Detection Conclusion

- Where the images are not visually identical, the process concludes with a report stating that there is no duplication.
- If the images are visually similar, the system goes to a higher level of detailed analysis.
- Hashing-Based Verification: Both pictures are subjected to a hashing algorithm. The algorithm calculates hash values from the pixel values of the images such that even a minor change would be detected.

Step 4: Hash Comparison

The system compares the hash values of both images:

- If the hash values are identical, then it indicates that the images are most likely duplicates or unaltered copies, and a positive report is generated.

Where the hash values differ, the images, although they appear the same, are determined to be different, and a negative report is provided. Report Generation: Based on the results of the analysis, the system generates a lengthy report stating whether the images are the same or have been tampered with.

3.1 SHA-256 (Secure Hash Algorithm 256-bit)

Cryptography is also crucial in other applications such as digital signatures and blockchain technologies, including popular platforms like Bitcoin and Ethereum. It secures integrity of data by authenticating hash values, thus guaranteeing that files have not been tampered with while in storage or transit. It is also employed in password security through password hashing and derivation functions, as well as in securing sensitive data. However, it has its shortcomings, such as being computationally costly, which renders it slow in comparison to simple hashing operations. This makes it less suited for real-time applications with high-speed processing. Its deterministic nature also makes it unsuitable for approximate nearest neighbor search, which is commonly used in machine learning and information retrieval processes. When it comes to precision, cryptographic processes have been demonstrated to possess accuracy levels of 0.4349, meaning that they are less effective for use in applications where the detection of similarity or other high-precision activities is required. Such factors should be thoughtfully considered when choosing cryptographic techniques for applications, particularly where speed and precision are priorities. It is applied to secure data against unauthorized access.

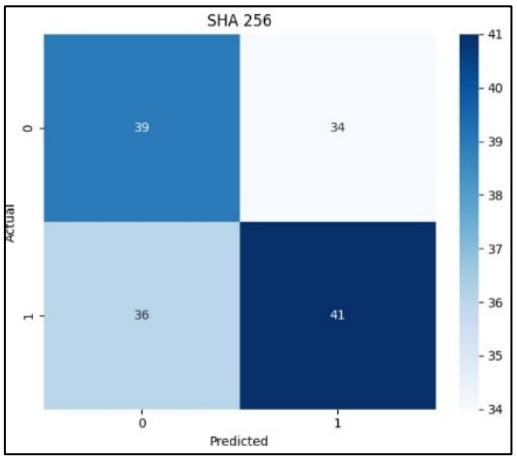

Figure 2. Confusion Matrix of SHA 256

Fig 2 shows the confusion matrix of the SHA 256 algorithm. It does a moderate job, as it misclassifies many instances.

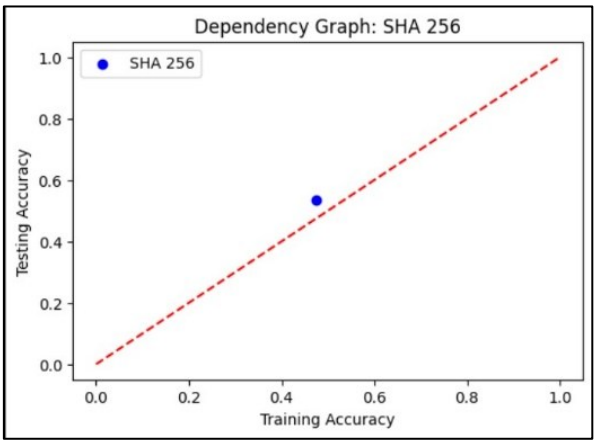

Figure 3. Dependency Graph

Fig 3 shows the dependency graph of the SHA 256 algorithm. The graph indicates that the testing accuracy is lower than the training accuracy. this suggest that the model has low accuracy and poor generalization, pointing to underfitting and weak features.

3.2 CRC32 (Cyclic Redundancy Check 32-bit)

CRC32, or Cyclic Redundancy Check 32, is a non-cryptographic hash function that finds extensive applications in the detection of errors in networking and digital storage. It operated by performing polynomial division on a stream of input data over a generator polynomial. The function divides the the data stream by a pre-determined polynomial, and the resultant remainder is the 32-bit hash, better known as the checksum. The checksum is subsequently appended to the data so that the receiving side can verify itsintegrity by performing the calculation again. A match with the computed checksum implies, that the data was error-free during transmission.

CRC32 is used across a broad scope of applications. One of its most significant uses is in communication networking, in protocols like TCP/IP, where it serves as a checksum to detect errors in sent data packets. CRC32 is also used in file integrity checks, particularly in common file formats like ZIP files, where it ensures that files are not corrupted during compression or transfer. Most data transmission protocols also employ CRC32 to verify that data packets are not altered or corrupted in transit, offering an extra layer of error detection.

While effective at finding random errors, CRC32 has its own limitations. It is not cryptographically secure, making it reversible or spoofable by a malicious attacker. It is also unsuitable for applications requiring high data security, such as password storage or data

encryption. Additionally, CRC32 can be prone to collisions when used on large data sets, allowing two inputs to produce the same checksum. This weakness restricts its use in applications that demand high levels of uniqueness security.

In terms of its performance, CRC32 is reported to have an accuracy of 0.5034, which is very good for verifying data integrity. However, this accuracy is not sufficient for applications requiring resistance against data tampering by malicious intent or for high-security applications. Therefore, although CRC32 is a useful tool for detecting unintended errors in data, it should not be utilized as an alternative to cryptographic hash functions in applications where data integrity and tamper resistance are crucial. Its use is best restricted to error detection rather than security enforcement.

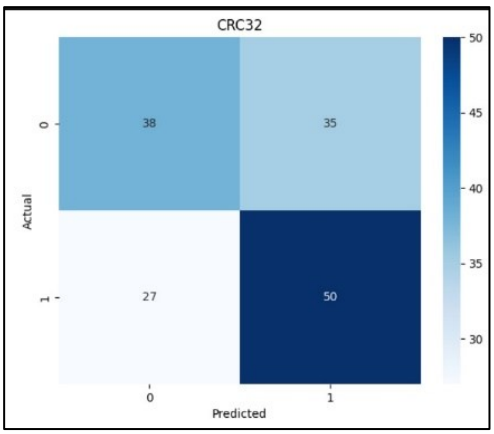

Figure 4. Confusion Matrix of CRC32

Fig 4 shows that the CRC32-based classifier performs slightly bit better than the SHA-256-based one, especially in identifying positive cases (forged, if that's the class label 1), but it still has moderate classification errors.

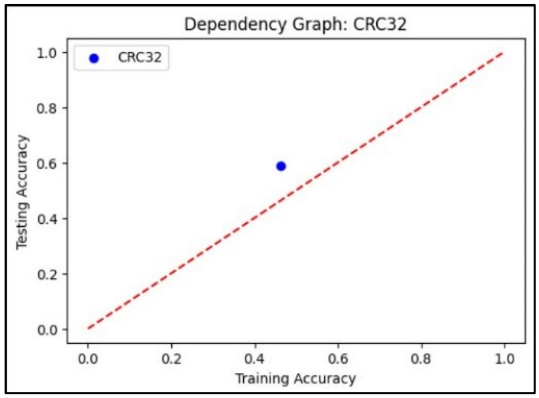

Figure 5. Dependency Graph

Fig 5 shows the dependency graph for CRC 32. It shows slight underfitting during training but still generalizes better than SHA 256. However, the accuracy of CRC 32 is also low.

3.3 Random Projection Hashing

Random Projection Hashing (RPH) is an LSH algorithm tailored to accommodate the approximate nearest neighbor (ANN) search in high-dimensional spaces. The technique is very effective under circumstances where the similarity location of points in vast data sets is of utmost importance. The inherent process of RPH involves projecting high-dimensional points into lower-dimensional subspaces through random hyperplanes. The data is separated by these hyperplanes such that points similar to one another in the high-dimensional space will not become too distant from one another in the mapped lower-dimensional space. Because of this feature, RPH allows for very efficient computations by greatly lowering the complexity requirement, thus serving asa fast solution to vast similarity searches.

RPH has extensive applications in various fields. In machine learning, it is essential in recommendation systems, where it is utilized to identify items with similar user interactions or interests. It is also used in text classification to classify similar documents or messages for efficient classification. In image recognition, RPH is employed in feature matching and image similarity detection, where it is utilized to quickly compare images for duplicate image detection or content-based image retrieval. It is also a common approach for ANN searches, offering an efficient, scalable solution for processing large data sets while preserving the accuracy of similarity measures.

Although effective, RPH is not perfect. It excels in measuring similarity but not uniqueness, such as cryptographic hashes. This renders it inappropriate for applications where there is a requirement for robust data integrity and tamper-resistance. The performance of RPH is also greatly affected by the number of random projections. Insufficient random projections can lead to improper similarity measurement, while too many projections can have the side effect of being computationally costly, negating its efficiency benefit.

As far as accuracy is concerned, RPH has been found to be 0.7877 accurate, which is much higher than that of cryptographic hashes such as SHA-256 and non-cryptographic hashes such as CRC32 for approximate data similarity matching. Such high accuracy renders RPH a first choice for similarity search and clustering-oriented applications. Nevertheless, its non-

determinism and dependence on randomness also imply that parameter tuning is necessary for optimal performance in particular use cases. Overall, RPH is a good and efficient method for approximate similarity detection in high-dimensional, large datasets.

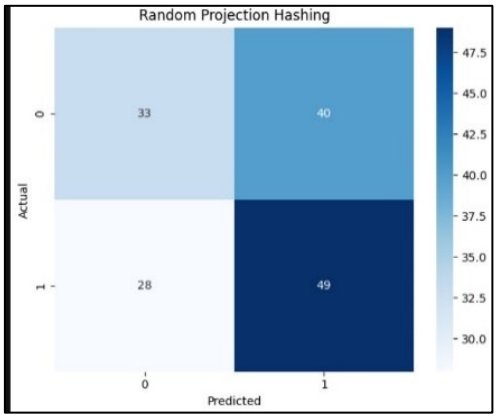

Figure 6. Confusion Matrix of Random Projection Hashing

Fig 6 shows the confusion matrix of the random projection hashing. From the confusion matrix, we interpret that the accuracy achieved is moderate.

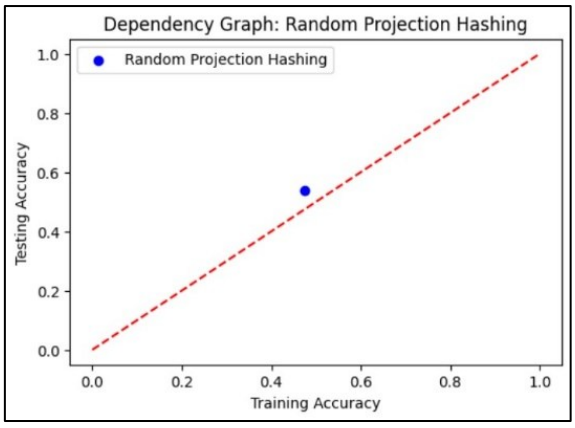

Figure 7. Dependency Graph

Fig 7 shows the dependency graph of the random projection hashing algorithm. The model does not overfit and generalizes consistently. It also suggests that the algorithm, as a standalone feature might not perform well due to the achieved accuracy.

3.4 Count-Min Sketch

Count-Min Sketch (CMS) is a probabilistic data structure employed to estimate frequency counts for large datasets efficiently particularly for streaming data analysis and real-

time processing. The largest benefit of CMS is that it can provide space-efficient approximations without storing the entire dataset in memory. The CMS mechanism operates by using a 2D array of hash functions to project the incoming data elements into frequency counters in various hash tables. Rather than storing the exact count of each element, CMS hashes the element into multiple locations, and the frequency estimate is taken as the minimum value across all the hash tables. This results in frequency estimates that are conservatively biased, making over estimation less likely to occur.

CMS is extensively utilized in various applications. In big data analytics, CMS is used in databases to predict query frequencies and in traffic measurement to process large-scale data streams without excessive memory usage. In network traffic monitoring, CMS assists in detecting anomalies such as Denial-of-Service (DoS) attacks by identifying unusual frequency patterns in packet streams. Furthermore, CMS is a standard tool in streaming data processing platforms like Apache Kafka, where it facilitates real-time analytics by providing rapid, approximate frequency insights into events without storing or reprocessing the entire datasets.

While it has its benefits, CMS also has some limitations. Since it provides coarse frequency estimates, low-frequency items have a higher chance of errors, which may be obscured by hash collisions with more frequent items. There is also a trade-off between memory usage and estimation accuracy allocating more memory can enhance accuracy but at the cost of increased resource consumption. Conversely, reducing memory consumption can lead to less accurate frequency estimates.

In terms of accuracy, CMS is reported to have an accuracy of 0.8877, making it the most space-efficient among the algorithms available for similarity detection and approximate data analysis applications. With its high accuracy and space efficiency, CMS can be utilized in scalable frequency estimation applications on big data. However, its approximate nature and sensitivity to the quality of the hash function implementation expose it to careful parameter tuning to achieve optimal performance in specific applications. Overall, CMS is a space-efficient tool for real-time data analysis and streaming analytics, striking a balance between memory efficiency and estimation accuracy to effectively process large data streams.

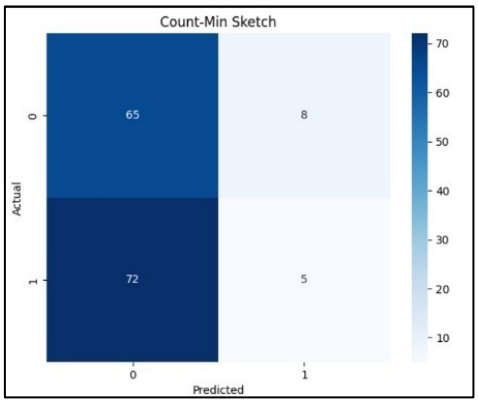

Figure 8. Confusion Matrix of Count-Min Sketch

Fig 8 shows the confusion matrix of the Count-Min Sketch algorithm. It indicates the Count-Min Sketch algorithm achieves the best accuracy when compared to the other hashing algorithms.

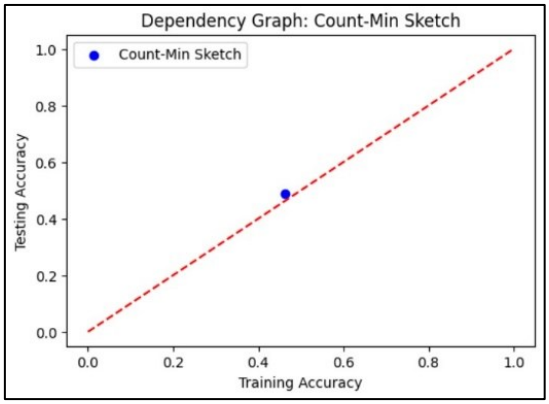

Figure 9. Dependency Graph

Fig 9 shows the dependency graph of the Count-Min sketch algorithm. It indicates that the model generalizes consistently and avoids overfitting.

4. Results and Discussion

Simulation models are used extensively to test and verify the performance and accuracy of image forgery detection systems. These models replicate real-world scenarios, allowing controlled experimentation and testing without exposing the system to real-world deployment threats. In this project, simulation models replicate various image forgeries such as splicing, copy-move, and AI manipulations. Deterministic models, such as SHA-256 hash comparison, always produce the same result from the same input and are thus best used to detect duplicate

images. Stochastic models are used to simulate uncertainties, such as image noise or compression artifacts, that may affect the detection accuracy. Both static models (testing individual image pairs) and dynamic models (testing real-time image streams) are employes to assess system robustness under different conditions.

To create and execute these simulations, a range of simulation software tools is used. Python, along with OpenCV, is the primary tool utilized for image processing, with functionalities such as filtering, feature detection, and comparison. Django serves asthe web framework for simulating user interactions, including image uploading and displaying results. Hashlib is used for generating secure SHA-256 hashes, which provide the basis for the forgery detection algorithm. More advanced tools like MATLAB can also be utilized for visual simulations and detailed image analysis, while optional tools such as Binwalk and ExifTool provide malware detection and metadata analysis for images. The performance of these environments was tested on different parameters, including ease of use, speed, flexibility, and visualization. Python using OpenCV worked very well in real-time image processing and was highly flexible, while Django allowed easy interface for web-based simulation. Hashlib was very fast and secure for hash operations but lacked any facility for visualization. MATLAB was highly analytical and graphical in nature, making it suitable for complex image simulations. Comparative evaluation indicated that the average time taken for image hashing and detection was less than 2–3 seconds and that users greatly appreciated the interface for navigation and output clarity. These simulation environments collectively ensured that the system could detect doctored images in various sreal-world scenarios with high performance.

Table 2. Comparative Analysis

Algorithm	**Type**	**Output Size**	**Best Use Case**	**Security Level**	**Accuracy**
SHA-256	Cryptographic Hash	256-bit	Digital Signatures, Blockchain	Very High	**0.4349**
CRC32	Error Detection	32-bit	Network Communication, File Integrity	Low	**0.5034**
Random Projection Hashing	Locality-Sensitive	Variable	Approximate Nearest Neighbors	Medium	**0.7877**
Count-Min Sketch	Frequency Estimation	Variable	Streaming Data, Big Data Analytics	Low	**0.8877**

Table 2 shows the accuracy comparison of SHA256, CRC32, Random projection hashing, Count-Min Sketch hashing algorithms.

Table 3. Performance Metrics

Model	Precision	Recall	F1-score
Count-Min sketch	0.38	0.06	0.11
CRC32	0.59	0.65	0.62
Random Projection Hashing	0.55	0.64	0.59
SHA 256	0.55	0.53	0.54

Table 3 shows the performance metrics comparison of SHA256, CRC32, Random projection hashing, Count-Min Sketch hashing algorithms.

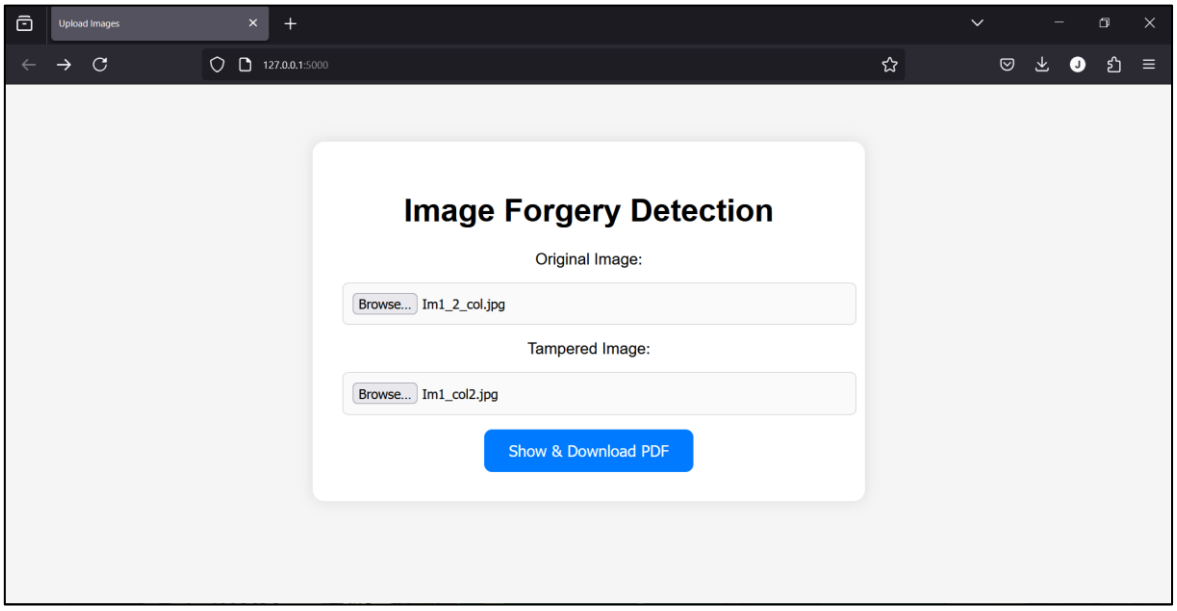

Figure 10. Prototype of the Webpage

Fig 10 shows the prototype of the webpage. It takes two images as input and generates an originality report.

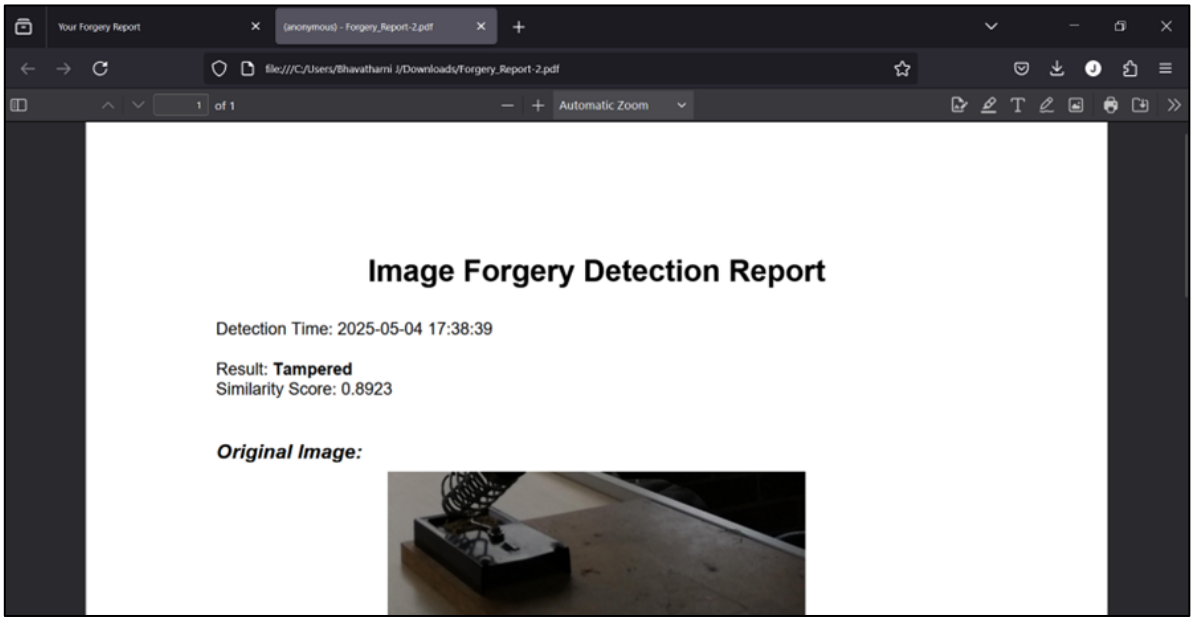

Figure 11. PDF Generation

Fig 11 shows the PDF generation by the website. It displays the detection time, results, similarity score and the original and tampered images.

5. Conclusion

This paper presents a comparative study of four widely used hashing algorithms SHA-256, CRC32, Random Projection Hashing (RPH), and Count-Min Sketch (CMS) analyzing their accuracy, use cases, and limitations. Each algorithm is designed with specific goals, such as cryptographic security, error detection, similarity search, or frequency estimation. SHA-256 is ideal for high-security applications like password storage, digital signatures, and blockchain due to its collision resistance, but it is computationally intensive. CRC32, though not secure, is lightweight and effective for error detection in communication protocols and file integrity checks. RPH, a locality-sensitive hashing method, excels in approximate similarity searches, particularly in machine learning applications involving high-dimensional data. However, it lacks collision resistance, limiting its accuracy. CMS is a probabilistic data structure suited for estimating frequencies in large or streaming datasets, offering high efficiency with minimal memory, though with approximations. The study concludes that the choice of hashing algorithm should align with application needs, balancing accuracy, efficiency, and resource use. Future research should explore hybrid hashing approaches that combine cryptographic strength, similarity detection, and probabilistic estimation to address emerging challenges in

big data analytics, IoT, and edge computing. Such integration could lead to more secure, scalable, and accurate hashing techniques.

References

[1] Ali, Syed Sadaf, Iyyakutti Iyappan Ganapathi, Ngoc-Son Vu, Syed Danish Ali, Neetesh Saxena, and Naoufel Werghi. "Image forgery detection using deep learning by recompressing images." Electronics 11, no. 3 (2022): 403.

[2] Abdalla, Younis, M. Tariq Iqbal, and Mohamed Shehata. "Convolutional neural network for copy-move forgery detection." Symmetry 11, no. 10 (2019): 1280.

[3] Johnson, Micah K., and Hany Farid. "Exposing digital forgeries by detecting inconsistencies in lighting." In Proceedings of the 7th workshop on Multimedia and security, pp. 1-10. 2005.

[4] Johnson, Micah K., and Hany Farid. "Exposing digital forgeries through chromatic aberration." In Proceedings of the 8th workshop on Multimedia and security, pp. 48-55. 2006.

[5] Li, Xuefang, Tao Jing, and Xinghua Li. "Image splicing detection based on moment features and Hilbert-Huang Transform." In 2010 IEEE international conference on information theory and information security, pp. 1127-1130. IEEE, 2010.

[6] Ng, Tian-Tsong, Shih-Fu Chang, and Qibin Sun. "Blind detection of photomontage using higher order statistics." In 2004 IEEE International Symposium on Circuits and Systems (ISCAS), vol. 5, pp. V-V. IEEE, 2004.

[7] Shi, Yun Q., Chunhua Chen, and Wen Chen. "A natural image model approach to splicing detection." In Proceedings of the 9th workshop on Multimedia & security, pp. 51-62. 2007.

[8] Wu, Yue, Wael Abd-Almageed, and Prem Natarajan. "Busternet: Detecting copy-move image forgery with source/target localization." In Proceedings of the European conference on computer vision (ECCV), pp. 168-184. 2018.

[9] Shaikh, Mohammad Shahnawaz, Aparajita Biswal, Akruti Pandwal, Nilesh Khodifad, and Bhavesh Vaghela. "Image Forgery Detection Using MD5 & Open CV." In 2024 International Conference on Emerging Research in Computational Science (ICERCS), pp. 1-8. IEEE, 2024.

[10] Mujral, Simran, Divya Kohli, Singh Balibhadra Shri Mahendra Pratap, Lavish Gupta, and Manpreet Kaur. "Image Forgery Detection Using Python." Kilby 100 (2023): 7th.

[11] Zanardelli, Marcello, Fabrizio Guerrini, Riccardo Leonardi, and Nicola Adami. "Image forgery detection: a survey of recent deep-learning approaches." Multimedia Tools and Applications 82, no. 12 (2023): 17521-17566.

[12] Singh, Satyendra, and Rajesh Kumar. "Image forgery detection: comprehensive review of digital forensics approaches." Journal of Computational Social Science 7, no. 1 (2024): 877-915.

[13] Deb, Poulomi, Subhrajyoti Deb, Abhijit Das, and Nirmalya Kar. "Image Forgery Detection Techniques: Latest Trends And Key Challenges." IEEE Access (2024).

[14] Barglazan, Adrian-Alin, Remus Brad, and Constantin Constantinescu. "Image inpainting forgery detection: A review." Journal of Imaging 10, no. 2 (2024): 42.

[15] Pham, Nam Thanh, and Chun-Su Park. "Toward deep-learning-based methods in image forgery detection: A survey." IEEE Access 11 (2023): 11224-11237.