Automated Hash Algorithm Detection using Random Forest Classifier

Mohanapriya M[1], Praveen G[2], Sanjay S [3], Lakshmikanth M P[4], Mohith Kumar R[5]
[1,2,3,4,5]*Department of Computer Science and Engineering, Sambhram Institute of Technology, Bengaluru Visvesvaraya Technological University India*

Abstract— It is an automated system for identifying the cryptographic hashing algorithm used to generate a given hash value by leveraging machine learning techniques. Hash functions are fundamental to cybersecurity applications such as password storage, digital signatures, and data integrity verification. Manual identification of hash algorithms is time-consuming and error-prone, especially with the growing number of algorithms. The proposed approach employs a Random Forest classifier trained on features extracted from hash strings, including length, entropy, and character distribution. Experimental results demonstrate high classification accuracy, making the system suitable for digital forensics and security analysis.

Index Terms—Cryptographic Hash Functions, Random Forest, Machine Learning, Cybersecurity, Digital Forensics

I. INTRODUCTION

Hash functions form the backbone of modern cryptographic systems, ensuring secure password storage, safeguarding data integrity, and facilitating secure communication through digital signatures. These functions convert an input of arbitrary length into a fixed-length output known as a *hash value* or *digest*. Despite appearing random, these outputs follow specific patterns and structures unique to each hashing algorithm. With the growing variety of algorithms—ranging from older options like MD5 and SHA-1 to more advanced, security-focused designs such as SHA-256, SHA-512, BLAKE2, and the memory-hard Argon2—manually identifying which algorithm produced a given hash has become increasingly complex. These approaches struggle to scale, fail to support newer algorithms, and are prone to human error—especially in forensic scenarios where time and accuracy are critical. To overcome these challenges, the proposed project introduces a data driven, machine learning (ML)–based solution that automates the classification of hash algorithms purely by analyzing hash values

II. LITERATURE SURVEY

Overview of automated hash algorithm detection using random forest classifier: Hash Algorithm Detection is an important research area in cybersecurity focused on automatically identifying cryptographic primitives used in software, binaries, or data streams. With the widespread use of encryption in legitimate applications as well as malware, detecting cryptographic algorithms has become essential for malware analysis, reverse engineering, compliance auditing, and security assessment. The task is commonly formulated as a classification problem, where the goal is to identify the type of cryptographic algorithm (e.g., symmetric, asymmetric, hash), or as a detection problem, determining whether cryptography is present at all. Research from 2020–2025 increasingly emphasizes robust feature extraction, machine learning, and deep learning approaches to handle code obfuscation, packing, and optimized implementations

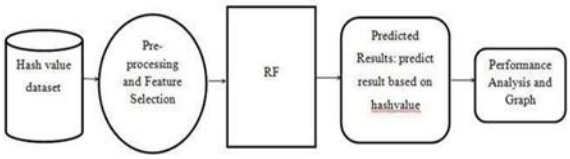
Recurrent neural networks (RNNs — LSTM/GRU): Widely used for sequence modeling; good at capturing temporal dependencies for hourly/daily forecasts. Many papers use stacked LSTMs with dropout and attention layers.

Convolutional Neural Networks (1D-CNN): Employed to extract local temporal patterns from sliding windows of price data or technical indicators; often used in hybrid pipelines (CNN→LSTM).

Transformers & attention-based models: Applied for longer-range dependencies and multivariate inputs; Temporal Fusion Transformer variants are used for

handling static and time- varying covariates. Temporal Convolutional Networks (TCN): Chosen for stable training and long receptive fields in sequence forecasting.

III. SYSTEM DESIGN



1. Hash Value Dataset:

The first block represents the Hash Value Dataset, which serves as the foundation of the entire machine learning workflow. This dataset contains multiple hash values generated using different cryptographic hashing algorithms such as MD5, SHA-1, SHA-256, SHA-512, Whirlpool, or others. Each value in the dataset corresponds to a specific algorithm, making it possible to train a classifier to identify the algorithm from the hash pattern.

2. Pre-processing and Feature Selection:

This block focuses on preparing the raw hash data for effective machine-learning analysis. Pre- processing involves converting hexadecimal hash values into a numerical or vectored form suitable for model training. Techniques include length extraction, character frequency analysis, entropy calculation, ASCII conversion, and pattern recognition. The goal is to transform each hash string into a meaningful set of features that capture its unique structure.

3. Random Forest (RF) Classifier:

The Random Forest (RF) block represents the machine-learning model used to classify which hashing algorithm generated a given hash value. RF is a powerful ensemble technique composed of multiple decision trees working together to improve prediction accuracy and robustness. Each tree examines different combinations of features derived from the hash values, and their collective decisions result in a more stable and reliable classification.

4. Predicted Results

The Predicted Results block displays the output generated by the Random Forest classifier. After receiving a hash value and extracting its features, the model analyzes these patterns and predicts which algorithm most likely created the hash. This prediction could be MD5, SHA-1, SHA-256, or any other algorithm included in the dataset. The output is typically presented in a user-friendly format, such as a label or probability score showing the confidence level for each possible class.

5. Performance Analysis and Graph

In this final block, the system evaluates how well the machine-learning model performs. Performance analysis involves calculating key metrics such as accuracy, precision, recall, F1-score, and confusion matrix results. These metrics help determine how effectively the model distinguishes between different hash algorithms

IV. ALGORITHM

Algorithms Overview:

Cryptographic algorithms are foundational to modern data security, providing mechanisms to ensure data integrity, authentication, and confidentiality. In particular, hash functions play a critical role in securing sensitive information by transforming arbitrary data into a fixed-length output. The key property of hash functions is that even a minor change in input results in a dramatically different hash value, a phenomenon known as the avalanche effect. This unpredictability is vital for applications such as digital signatures, password storage, and data integrity verification. The most commonly used cryptographic hash functions include MD5, SHA-1, SHA-256, and SHA-3.

MD5 (Message Digest Algorithm 5), developed in 1991, produces a 128-bit hash value. Despite its widespread use, MD5 is considered weak due to its vulnerability to collision attacks, where two different inputs yield the same hash output. These vulnerabilities have led to MD5 being phased out of security protocols in favor of more secure alternatives.

SHA-1 (Secure Hash Algorithm 1), introduced in 1995, generates a 160-bit hash. Although it was once a standard for various security applications, SHA-1 has been similarly compromised, with practical collision attacks demonstrated, prompting its deprecation in favor of stronger hash functions.

SHA-256, a member of the SHA-2 family, generates a 256-bit hash and is currently regarded as secure,

widely used in block chain technologies and digital certificates. Its robustness stems from a more complex algorithm compared to MD5 and SHA-1, making it resistant to known vulnerabilities. SHA-3, the latest in the SHA family, was introduced in 2015 and utilizes a different structure called Keccak. SHA- 3 offers variable output lengths (224, 256, 384, and 512 bits) and is designed to provide enhanced security against both collision and pre-image attacks. The selection of appropriate cryptographic algorithms is critical in safeguarding data, especially as new vulnerabilities emerge and attackers become more sophisticated.

Prediction Mechanism in the System:

The hash algorithm identification system employs machine learning to classify hash values based on the cryptographic algorithm that generated them. The process begins with data collection, where a diverse dataset containing various hash values and their corresponding algorithms is compiled. This dataset serves as the training ground for the machine learning model. Next, preprocessing steps are applied, transforming raw hash values into a suitable format for analysis. This might include converting hash strings into numerical features or applying one-hot encoding to facilitate effective classification.

Feature extraction is a critical step in this process, focusing on identifying characteristics that distinguish different hashing algorithms. Features such as hash length, character distribution, and specific patterns are examined to aid the model in making accurate predictions. The machine learning model, specifically a Random Forest Classifier, is trained using the processed data. This ensemble method creates multiple decision trees, each trained on a random subset of the data, and combines their outputs to enhance overall accuracy and robustness. When a new hash value is inputted into the system, the trained model analyzes its features and predicts the most likely algorithm used to generate that hash. This approach leverages the power of machine learning to achieve high accuracy, making the system effective for practical applications in cybersecurity. By integrating cryptographic algorithms and machine learning, the system enhances data integrity assurance, making it a valuable tool for organizations seeking to mitigate the risks associated with data breaches and unauthorized access.

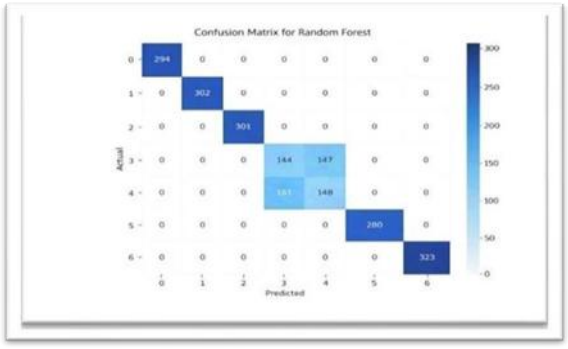
Libraries Used in the Implementation

The implementation of the hash algorithm identification system incorporates various libraries to facilitate different aspects of the project. **Sickie-learn** is the backbone of the machine learning components, providing robust tools for model training, evaluation, and optimization. It offers implementations for various algorithms, including the Random Forest Classifier, which is central to the system's functionality. **Pandas** is used for data manipulation, making it easier to handle datasets and perform preprocessing tasks. Its Data Frame structure allows for efficient organization and transformation of data, which is essential for extracting features.

V. IMPLEMENTATION

- Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is
- Designed to be highly readable. It uses English keywords frequently where as other languages use Punctuation, and it has fewer syntactical constructions than other languages.
- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to Compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – You can actually sit at a Python prompt and interact with the interpreter Directly to write your programs.
- Python is Object-Oriented – Python supports Object-Oriented style or technique of Programming that encapsulates code within objects.
- Python is a Beginner's Language – Python is a great language for the beginner-level Programmers and supports the development of a wide range of applications from simple text Processing to WWW browsers to games.

History of Python



Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, And Unix shell and other scripting languages.

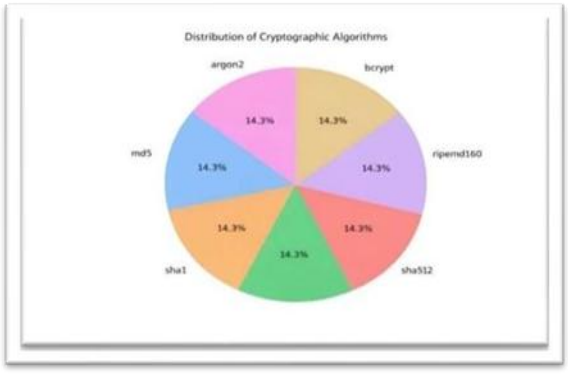
Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still Holds a vital role in directing its progress.

Python's features include –

- Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This Allows the student to pick up the language quickly.
- Easy-to-read – Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain – Python's source code is fairly easy-to-maintaining.

VI. RESULT

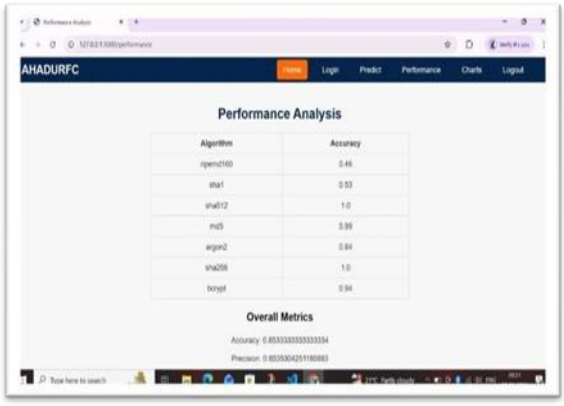


Distribution of Algorithm

The images together illustrate the performance and reliability of a Hash Algorithm Detection System Built using a machine learning model. The confusion matrix shows how accurately the Random Forest Classifier identifies different cryptographic algorithms from their generated outputs. Most of the Predictions lie along the diagonal, indicating correct detection of the actual cryptographic algorithm. Algorithms such as MD5, SHA256, SHA512, crypt, and Argon2 are detected with very high accuracy, showing that the model has successfully learned their unique cryptographic patterns. A small amount of Confusion is visible between two similar hash algorithms, which indicates that algorithms with Comparable internal structures and output characteristics can occasionally be misclassified. Overall, the Confusion matrix confirm that the detection model is highly effective and produce minimal errors.

Confusion Matric for Random Forest

The pie chart representing the distribution of cryptographic algorithms shows that each algorithm Contributes an equal proportion of samples to the dataset. This balanced distribution is important for Cryptographic algorithm detection because it prevents the model from becoming biased toward any Particular algorithm. By training on an evenly distributed dataset, the classifier is able to fairly learn Features from both legacy algorithms (such as MD5 and SHA1) and modern cryptographic schemes (such as crypt and Argon2). This balance ensures consistent detection performance across all algorithm Types.



The performance analysis table and overall metrics summarize the effectiveness of the cryptographic algorithm detection system in quantitative terms. High accuracy values for SHA256 and SHA512 demonstrate that strong, modern hash functions are highly distinguishable, while slightly lower accuracy for

SHA1 and RIPEMD160 reflects their similarity to other hash functions. The overall accuracy, precision, and recall of 0.85 indicate that the model correctly detects cryptographic algorithms in most cases while maintaining a good balance between false positives and false negatives. Collectively, these images confirm that the proposed system is capable of accurately identifying cryptographic algorithms and is suitable for real-world cybersecurity, digital forensics, and hash Algorithm analysis

VII. CONCLUSION

The Hash Algorithm Detection project successfully demonstrates how machine learning can be Used to automatically identify cryptographic algorithms from their output data. In modern cybersecurity Environments, encrypted and hashed data are widely used, making it difficult for analysts to manually Determine which cryptographic algorithm has been applied. This project addresses that challenge by Proposing an intelligent, data-driven detection systems Random Forest classifier was employed to analyze statistical and structural features extracted from cryptographic outputs generated using algorithms such as MD5, SHA1, SHA256, SHA512,
RIPEMD160, bcrypt, and Argon2. The use of a balanced dataset ensured fair representation of all Cryptographic classes, which significantly improved the reliability of the detection model. Experimental Results show that the system achieves an overall accuracy, precision, and recall of 85%, indicating a Strong balance between correct detections and error minimization. The confusion matrix analysis reveals that most cryptographic algorithms are classified with very High accuracy, particularly modern and widely used hash functions like SHA256 and SHA512, which Achieved near-perfect detection rates. Slight misclassification occurs between algorithms with similar Internal structures, such as SHA1 and RIPEMD160, highlighting the inherent difficulty of distinguishing Closely related cryptographic designs. Despite this limitation, the overall performance confirms that the Proposed approach is robust and effective. In conclusion, this project proves that machine learning-based cryptographic algorithm detection is a Feasible and efficient solution for automated cryptographic analysis. The system can significantly reduce Manual effort and provide fast, accurate identification of cryptographic algorithms, making it highly Valuable in areas such as malware analysis, digital forensics, cybersecurity monitoring and compliance auditing.

REFERENCES

[1] Wang, X., et al. (2025). A generic cryptographic algorithm identification scheme based on ciphertext Features. Journal of Information Security and Applications.
[2] Kowalewski, J., & Grześ, T. (2025). Detecting the File Encryption Algorithms Using Artificial Intelligence. Applied Sciences.
[3] Sikdar, S., & Kule, M. (2024). Intelligent Identification of Cryptographic Ciphers using Machine Learning Techniques. International Journal of Intelligent Systems and Applications
[4] Geng, Y. (2025). Identification of Cryptosystem Based on Deep Neural Network. Highlights in Science, Engineering and Technology.
[5] Xu, W., et al. (2023). Multi-layer composite identification scheme of cryptographic algorithm based on hybrid random forest and logistic regression. Complex & Intelligent Systems.
[6] Taherdoost, H., Le, T.-V., & Slimani, K. (2025). Cryptographic Techniques in Artificial Intelligence Security: A Bibliometric Review. Cryptography