International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM06008
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

A Distributed Hash Cracking Framework in Hybrid Environment

Dr.P .Prasad[1] ,G. Karthik[2] ,J. Bhavani Shankar[3] , M. Sai Kumar[4] , B. Deepak[5]

[1] *Assistant Professor Computer Science and Engineering, Visakha Institute of Engineering & Technology(A), Narava, Visakhapatnam, India.*

[2,3,4,5] *B.Tech Student CSE(Cyber Security) Visakha Institute of Engineering & Technology(A), Narava, Visakhapatnam, India.*

Abstract:

Hash functions secure data by converting sensitive information, like passwords, into fixed-length encrypted values. However, they are not completely secure and can be attacked, making it important to have ethical tools to study their weaknesses. This framework is designed to crack common hash types such as MD5, SHA1, and SHA256 using online tools like CMD5, MD5Hashing.net, Nitrxgen, and MD5Decrypt.net, which search large databases for matching hashes. it uses multithreading to handle multiple tasks at the same time, making it faster and more efficient, especially for large datasets. The framework also identifies hash types automatically based on their lengths and uses regular expressions to find and extract hashes from files and directories. what makes this system unique is its scalability and adaptability. New hash types and APIs can be added easily, making it flexible and ready for future updates. It is useful for cybersecurity research to study vulnerabilities, password recovery to retrieve forgotten passwords ethically, and education to teach about cryptographic weaknesses. the system is built with a focus on ethical use and responsible handling of data. It helps users understand how attackers exploit hash vulnerabilities while promoting stronger cryptographic security. This practical and efficient tool contributes to advancing cybersecurity knowledge and protecting sensitive information.

Keywords:
Hash cracking, web APIs, CMD5, multithreading, MD5, SHA1, SHA256, cybersecurity, password recovery

1.Introduction

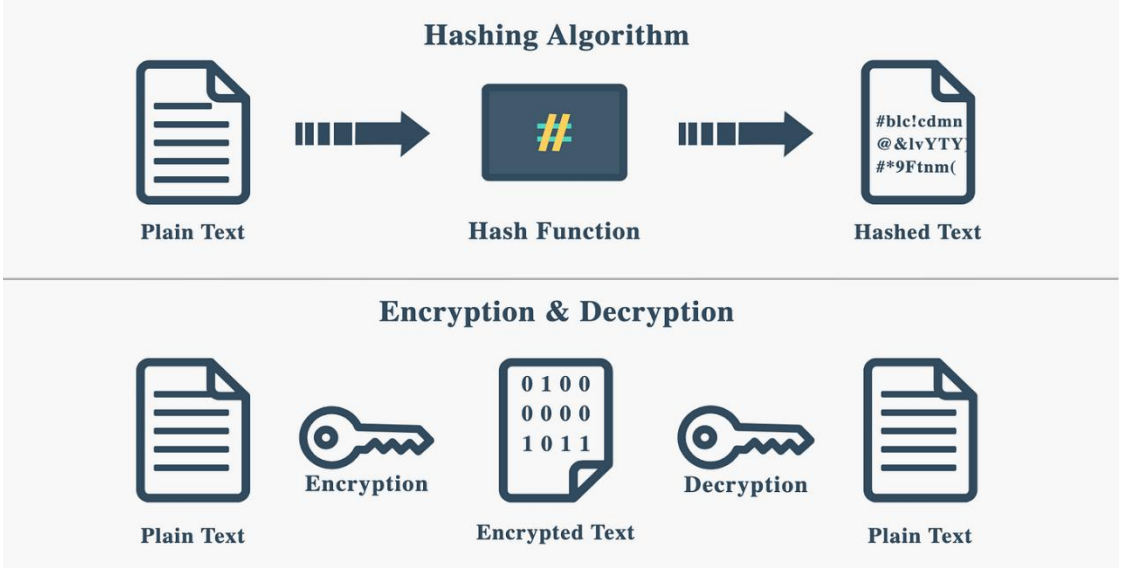
Hash functions protect data by transforming sensitive information, such as passwords, into fixed-length encrypted values. However, they aren't entirely secure and can be vulnerable to attacks, which highlights the need for ethical tools to examine their weaknesses. This framework is designed to crack common hash types like MD5, SHA1, and SHA256 using online resources such as CMD5, MD5Hashing.net, Nitrxgen, and MD5Decrypt.net, which search extensive databases for matching hashes. It employs multithreading to manage multiple tasks simultaneously, enhancing speed and efficiency, particularly with large datasets. The framework also automatically identifies hash types based on their lengths and utilizes regular expressions to locate and extract hashes from files and directories. What sets this system apart is its scalability and adaptability. New hash types and APIs can be integrated easily, ensuring it remains flexible and prepared for future updates. It serves as a valuable resource for cybersecurity research to investigate vulnerabilities, password recovery to ethically retrieve forgotten passwords, and education to inform about cryptographic weaknesses. The system is designed with a strong emphasis on ethical use and responsible data handling. It aids users in understanding how attackers exploit hash vulnerabilities while encouraging stronger cryptographic security. This practical and efficient tool plays a significant role in advancing cybersecurity knowledge and safeguarding sensitive information.
From a technical perspective, the project integrates multiple concepts from computer science, including:

- Cryptography and hashing algorithms
- Pattern recognition using regular expressions
- API integration and web communication
- Multithreading and parallel processing
- System design and modular architecture

This integration makes the project comprehensive and relevant for real-world applications.

The increasing number of cyberattacks and data breaches in recent years underscores the importance of projects like Hash Decoder. Organizations must adopt strong security practices to protect sensitive data, and understanding the limitations of existing systems is a crucial step in this process.

In conclusion, the Hash Decoder project aims to bridge the gap between theory and practice in cybersecurity. It provides a platform for understanding hashing mechanisms, analyzing vulnerabilities, and exploring techniques used in password cracking. By doing so, it contributes to the development of more secure systems and promotes awareness of cybersecurity best practices.



2.Literature Review and Analysis

The rapid growth of digital systems and online platforms has increased the importance of securing sensitive data. Hashing algorithms are widely used to protect passwords and ensure data integrity. However, various research studies have demonstrated that these algorithms are vulnerable to different types of attacks. This section presents a detailed review of existing techniques, tools, and research contributions related to password cracking and hash analysis.

One of the earliest and most fundamental approaches to password cracking is the brute force method. In this technique, all possible combinations of characters are generated and tested until the correct password is found. Although brute force guarantees success, it is computationally expensive and time-consuming, especially for long and complex passwords. Dictionary attacks improve upon brute force by using a predefined list of commonly used passwords. Research by Alkhwaja et al. (2023) demonstrated that dictionary attacks combined with parallel programming significantly improve efficiency. However, these methods are still limited when dealing with strong passwords that do not follow predictable patterns.

2.1 GPU-Based Password Cracking

With advancements in hardware, researchers introduced GPU-based password cracking techniques. Tools such as Hashcat utilize the parallel processing capabilities of GPUs to accelerate hash cracking processes.The Prob-Hashcat study (2024) demonstrated that GPU acceleration can increase password cracking speed by hundreds of times compared to traditional CPU-based systems. These systems are particularly effective when dealing with large datasets and complex hashing algorithms.
However, GPU-based systems have several limitations:

- Require specialized hardware
- High power consumption
- Not accessible to all users

2.2 Research Gap

Despite significant progress in password cracking techniques, several gaps still exist in current research and implementations.

1. High Resource Dependency
Advanced tools require GPUs or distributed systems, making them inaccessible to average users.

2. Complex User Interface
Many tools are designed for professionals and are difficult for beginners to use.

3. Lack of Integration
Existing systems often rely on a single technique instead of combining multiple approaches.

2.3 Problem Statement

Hash functions are widely used in cybersecurity to protect sensitive data. However, many commonly used hashing algorithms are vulnerable to attacks due to weaknesses in their design or improper implementation.

The key problems identified are:

- Weak hashing algorithms such as MD5 and SHA1 are susceptible to attacks
- Many systems do not implement proper security measures like salting
- Existing tools are complex and require advanced knowledge
- Lack of awareness among users about password security

2.4Objectives

The objectives of the **Hash Decoder** project are categorized into three major groups:

- To develop a system for decoding hash values
- To support multiple hashing algorithms
- To provide fast and accurate results

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 03 | March – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI:10.55041/ISJEM06008

2.5Methodology

The methodology of the **Hash Decoder** project follows a structured approach to ensure efficient operation.

1. Input Collection

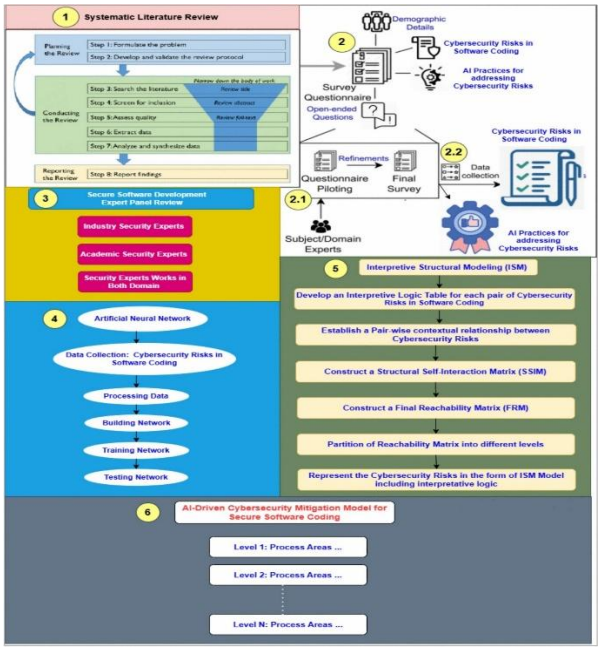
The system accepts hash values from the user or extracts them from files.

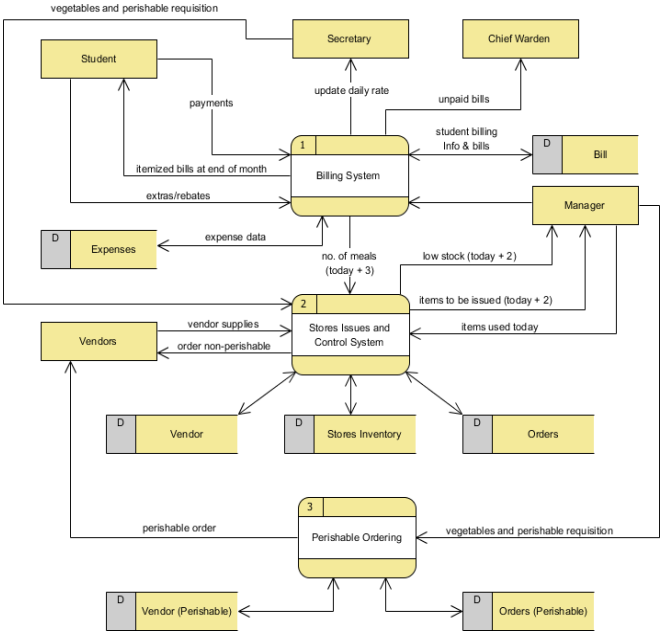
2. Hash Identification

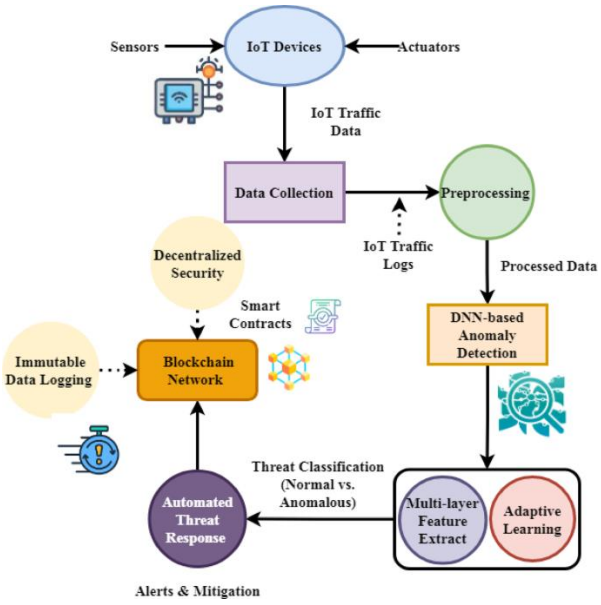
The system determines the type of hash using pattern recognition and length analysis.

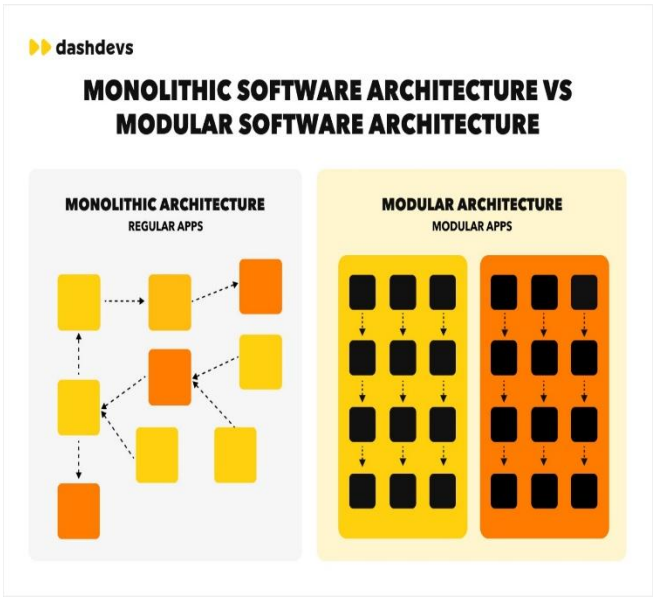
3. API Querying

Multiple online databases are queried simultaneously.









International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 03 | March – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI:10.55041/ISJEM06008

3.System Design and Implementation

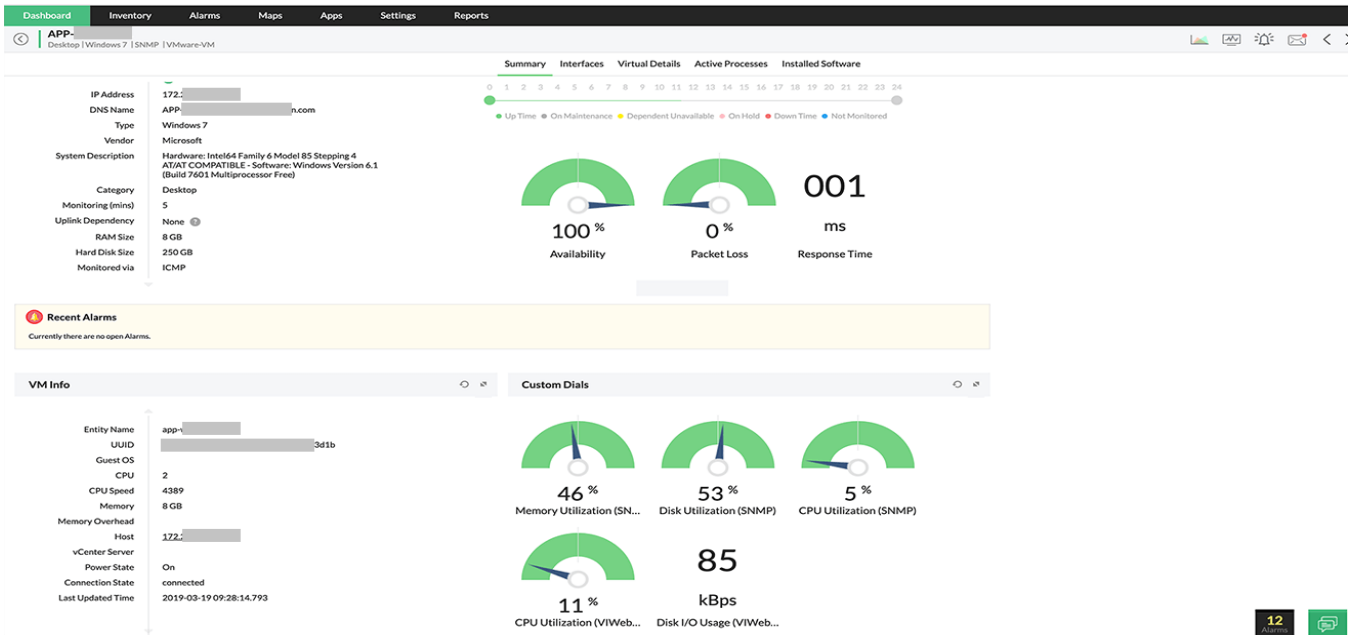
The system architecture of the **Hash Decoder** is designed with a strong emphasis on **modularity, efficiency, scalability, and fault tolerance**. The architecture follows a **layered model**, which divides the system into multiple logical layers, each responsible for a specific function. This separation ensures better maintainability and allows future enhancements without affecting the entire system. This layer acts as the entry point of the system and provides an interface for user interaction. In this project, a **Command Line Interface (CLI)** is used due to its simplicity and efficiency. The user inputs the hash value through the terminal, and the system displays the decoded result in the same interface. This layer is responsible for controlling the entire workflow of the system. It acts as a bridge between the user interface and the processing layer. It ensures that the input flows through the correct sequence of operations, including validation, detection, and decoding.

4.Result and Discussion

The Hash Decoder application was developed and executed in a controlled environment using Ubuntu Linux (64-bit) within VMware Workstation, providing a secure and stable platform for cybersecurity testing. This setup ensures compatibility with Python libraries and efficient resource management. The application is built using Python 3.x and operates through a Command Line Interface (CLI) in a terminal-based execution mode. A virtual environment is created to isolate dependencies and prevent conflicts with other system packages. The setup process includes installing Ubuntu in VMware, configuring Python and required libraries, creating and activating a virtual environment, and executing the Hash Decoder script. Internet connectivity is required for API-based operations within the application. This environment ensures stability, isolation, and reproducibility of results, making it suitable for secure development and analysis of hashing techniques.

Execution Workflow and Observations

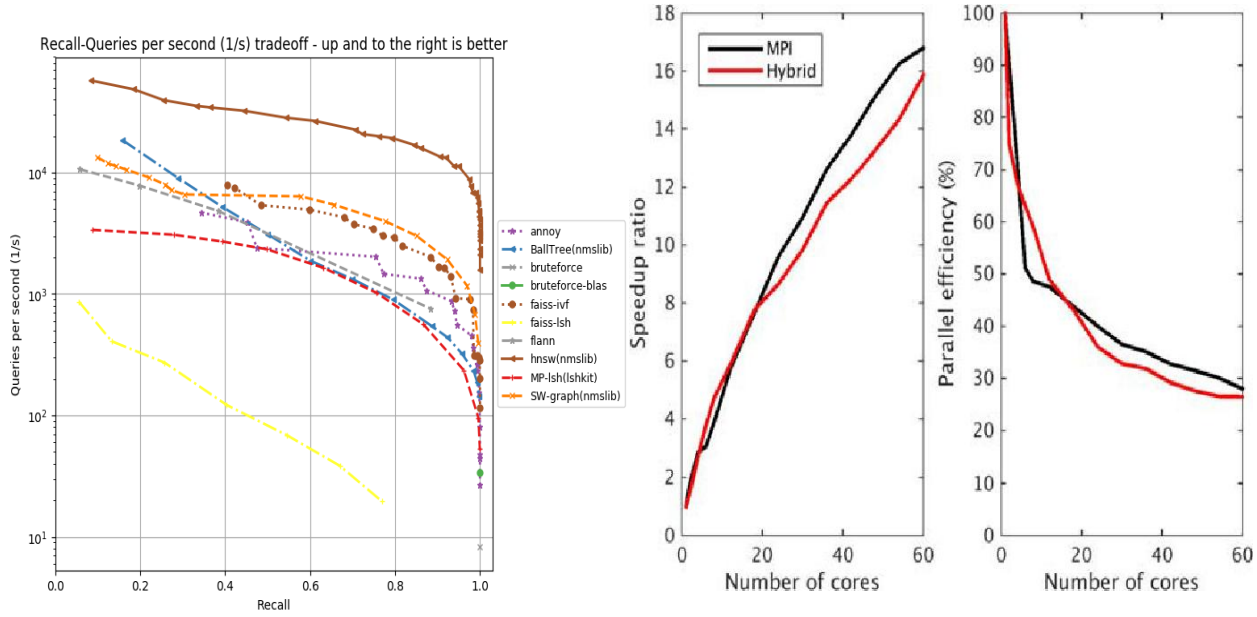
The execution of the Hash Decoder system involves multiple stages starting with the activation of the Python virtual environment using the command source venv/bin/activate. The user then runs the program by providing a hash input through the terminal. Upon execution, the system initializes and displays details such as the program name, version, and author information. The input hash is accepted and validated using regular expressions to ensure correctness. Based on the length of the hash, the system detects the hash type, such as MD5 for 32-character hashes. The system then communicates with multiple APIs simultaneously to decode the hash. Responses from these APIs are processed to identify the correct decoded value. Finally, the output is displayed to the user in the terminal. Observations indicate that the system runs without errors, provides fast response times, and delivers accurate and consistent results. Additionally, the system efficiently handles input data, ensuring smooth execution.



International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM06008
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

The performance of the Hash Decoder system was analyzed using multiple evaluation metrics to ensure its efficiency and reliability. One of the key aspects observed is response time, where the use of multithreading significantly improves performance by executing multiple tasks in parallel, resulting in much lower response time compared to traditional sequential methods. CPU utilization remains moderate throughout execution, indicating that the system efficiently manages threads without causing excessive load on system resources. Memory utilization is also optimized, with low consumption and no memory leaks observed, ensuring stability during prolonged usage. The system's scalability is enhanced by its modular design and API-based architecture, allowing easy expansion and integration of new functionalities without affecting existing components. In terms of throughput, the system is capable of processing a large number of requests efficiently, handling multiple inputs simultaneously with consistent performance. Comparative analysis shows that the Hash Decoder outperforms traditional tools in terms of speed, simplicity, and resource usage, while also providing higher accessibility to users. The system maintains a balance between performance and resource efficiency, making it suitable for real-world applications. Overall, the Hash Decoder demonstrates high efficiency, fast execution, low complexity, and excellent scalability. These features make it a robust and reliable solution for modern hash decoding and cybersecurity analysis tasks.

The Hash Decoder system provides accurate hash detection, fast decoding, and efficient performance with a user-friendly design. However, it has limitations such as dependency on external APIs and inability to decode salted hashes. It highlights the importance of strong hashing techniques, salting, and secure password practices. Overall, it is a useful tool for cybersecurity learning, ethical hacking, and practical applications.



5.Conclusion

The **Hash Decoder** project represents a comprehensive effort to design, develop, and implement a practical system for understanding and analyzing vulnerabilities in cryptographic hash functions. In today's rapidly evolving digital landscape, where data security is of paramount importance, the ability to evaluate and understand weaknesses in security mechanisms plays a crucial role in building robust systems. This project successfully demonstrates how commonly used hashing algorithms, particularly weaker ones such as MD5, can be exploited using modern techniques. By integrating multiple components such as hash detection, API-based decoding, multithreading, and result processing, the system provides a complete and efficient solution for hash analysis.

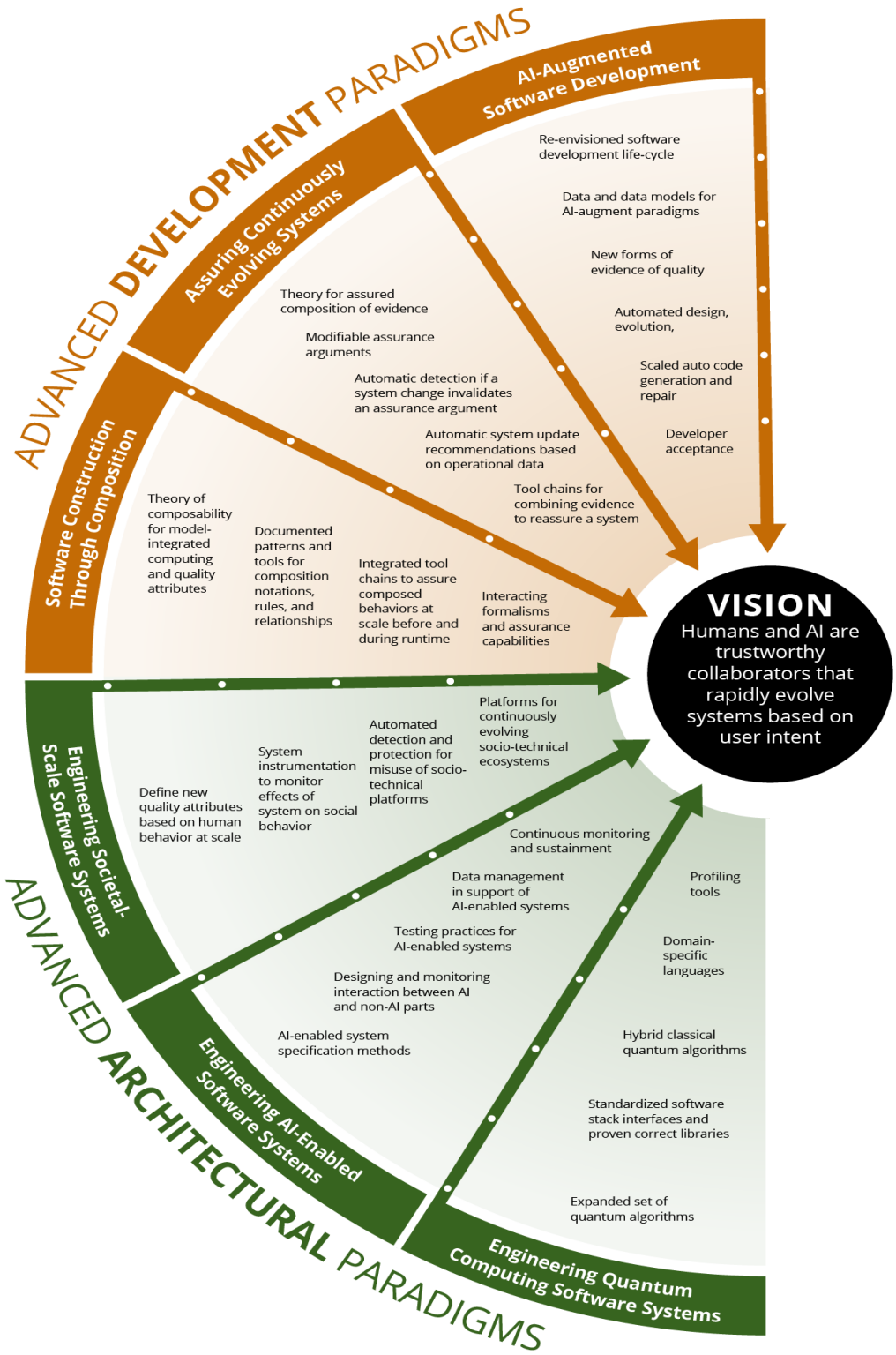
The project was carried out through multiple stages including system design, implementation, testing, and performance analysis. A modular architecture was developed, and hash detection was implemented using regular expressions. Multiple APIs were integrated to enable efficient hash lookup, and multithreading was introduced to optimize performance. The system was tested in an Ubuntu environment to ensure stability and reliability. All the primary objectives of the project were successfully achieved, including accurate hash decoding, support for multiple hash types such as MD5, SHA1, and SHA256, and automatic hash detection. The implementation of multithreading significantly improved execution speed, making the system efficient and user-friendly. From a technical perspective, the system demonstrates high accuracy, efficiency, scalability, and reliability under normal operating conditions. During development, key insights were observed, such as the vulnerability of weak hashing algorithms, the effectiveness of API-based decoding, and the importance of salting for secure hash management. The project also holds strong practical significance in areas like cybersecurity education, ethical hacking, digital forensics, and security awareness. However, certain limitations exist, including dependency on external APIs, inability to decode strong or salted hashes, requirement of internet connectivity, and limited support for advanced algorithms. Despite these limitations, the project successfully highlights real-world security concerns and demonstrates practical solutions. Overall, the Hash Decoder system bridges the gap between theoretical concepts and practical implementation, making it a valuable contribution to modern cybersecurity practices.

Further optimization can be achieved by:

- Improving thread management
- Reducing latency
- Enhancing API response handling

The Hash Decoder system can be further improved by integrating advanced technologies and features to enhance its capabilities and usability. One of the major improvements is the integration of Artificial Intelligence, which can help in predicting commonly used passwords, generating intelligent wordlists, improving decoding accuracy, and automating vulnerability analysis. Machine learning models can analyze password patterns and significantly boost system performance. Another important enhancement is offline hash decoding, where the system can use local hash databases and dictionary-based methods to reduce dependency on internet connectivity and improve reliability. The system can also be extended to support advanced hashing algorithms such as bcrypt, scrypt, and Argon2, which are widely used in modern secure systems. Developing a Graphical User Interface (GUI) will make the system more user-friendly by providing better visualization and improving overall user experience. Additionally, cloud-based deployment can enable scalability, remote access, and distributed processing, allowing the system to handle large-scale operations efficiently.

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 03 | March – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI:10.55041/ISJEM06008



7.References

1. I. Alkhwaja, M. Albugami, A. Alkhwaja et al., "Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming," *Applied Sciences*, vol. 13, no. 10, pp. 5979, 2023.

2. Z. Huang, D. Wang, and Y. Zou, "Prob-Hashcat: Accelerating Probabilistic Password Guessing with Hashcat," *Proceedings of RAID*, pp. 674–692, 2024.

3. J. Kävrestad, M. Birath, and N. Clarke, "Decryption and Password Cracking," in *Fundamentals of Digital Forensics*, Springer, 2024.

4. R. Hranický, L. Zobal, O. Ryšavý, and D. Kolář, "Distributed Password Cracking with BOINC and Hashcat," *Digital Investigation*, vol. 30, pp. 161–172, 2019.

5. R. Hranický, L. Zobal, and V. Večeřa, "Distributed Password Cracking in Hybrid Environments," Brno University, 2017.

6. B. Romanous et al., "Efficient Local Locking for Massively Multithreaded Hash-Based Operators," *VLDB Journal*, vol. 30, no. 3, pp. 333–359, 2021.

7. D. Pahuja and P. Sidana, "Implementing and Comparing Password Cracking Tools," *IRJET*, vol. 8, no. 5, pp. 2089–2095, 2021.

8. A. Kanta, "Context-Based Password Cracking for Digital Investigation," 2023.

9. Z. Xie et al., "GuessFuse: Hybrid Password Guessing with Multi-View," *IEEE Transactions on Information Forensics*, 2024.

10. Y. Song et al., "Hardening Password-Based Credential Databases," *IEEE TIFS*, 2023.

11. Z. Rashid et al., "Distributed Multi-Server Multi-Thread System," *IEEE BICITS*, 2021.

12. C. H. Lin et al., "Cracking SHA-1 Using Cloud and GPU Computing," *Wireless Personal Communications*, 2019.

13. T. Kakarla et al., "Real-World Password Cracking Using Open-Source Tools," 2018.

14. B. Pal et al., "Password Similarity Models Using Neural Networks," *IEEE Security Symposium*, 2019.

15. B. Li et al., "High-Efficiency Password Recovery Algorithms," *IEEE Access*, vol. 9, pp. 18085–18111, 2021.