1. **Introduction**
2. **Literature Review**
3. **Methodology**

In the pursuit of a comprehensive exploration of hashing techniques, the selection of algorithms for investigation was meticulously guided by a set of well-defined criteria. The criteria were designed to ensure a representative sampling across a spectrum of hashing approaches, thereby enriching the empirical foundation of the research. Eight distinct algorithms were judiciously chosen, each contributing unique attributes to the intricate tableau of the Graph With Table. Fundamental to this selection was the criterion of "Prominence and Historical Significance," whereby algorithms with a venerable history of deployment were deliberately included. This encompassed venerable stalwarts of the field such as SHA-1, MD5, and CRC32b, facilitating an insightful illustration of the evolutionary trajectory of hashing techniques over time.

Moreover, the criterion of "Diversity" played a pivotal role in shaping the selection framework. Recognizing the heterogeneous nature of the hashing landscape, the chosen algorithms spanned a gamut of properties, embracing both checksum generators such as Adler-32 and CRC32b, as well as cryptographic hash functions like MD5 and SHA-1. This meticulous curation ensures a nuanced examination of the myriad facets inherent in hashing methodologies, allowing for a holistic understanding of the diverse roles these algorithms play in the broader computational milieu. The research, therefore, embarks upon an in-depth analysis grounded in the judicious selection of algorithms, strategically aligning historical significance and diversity to foster a robust and insightful exploration of hashing techniques.

1. **Dataset Preparation and Pre-Processing**

**Data Collection:** To gather data, a prototype system that was carefully constructed with a combination of HTML, CSS, JavaScript, PHP, and MySQL was created. This prototype allowed for controlled experimentation and data collection by acting as the testing environment for the chosen hashing algorithms. Using a prototype system has a number of benefits, such as data management flexibility, real-time monitoring, and the capacity to evaluate the algorithms methodically in different scenarios.

**Experimental Design:** Carefully considered, the experimental design was created to meet the main goals of the study and evaluate the attributes. It required a sequence of deliberate actions:

**Dataset Selection**: To accurately reflect a range of real-world circumstances, a variety of input data sets were carefully selected. The size, substance, and complexity of these data sets varied, allowing for a thorough assessment of algorithm performance.

**Dataset Processing:** Every hashing method that was being considered was applied with great care to the chosen data sets in the prototype system. For each piece of data in the sets, hash values had to be generated in this stage. To enable in-depth examination, the resulting hash values were painstakingly captured and arranged. By maintaining records, it was made sure that the process of evaluating attributes was based on actual evidence.

**Quantitative Analysis and Attribute Evaluation:** A quantitative method was chosen in order to keep the assessment rigorous. Objective evaluation was made possible by the assignment of the attribute scores, which ranged from 1 to 5, according to predetermined criteria. To ascertain the qualities of the algorithms, a thorough analysis of the recorded hash values was conducted. Every attribute was subjected to assessment standards, which included security, correctness, authenticity, and efficiency. These standards were created in order to measure each algorithm's performance in a uniform way.

1. **Experimental Attributes and Ratings**

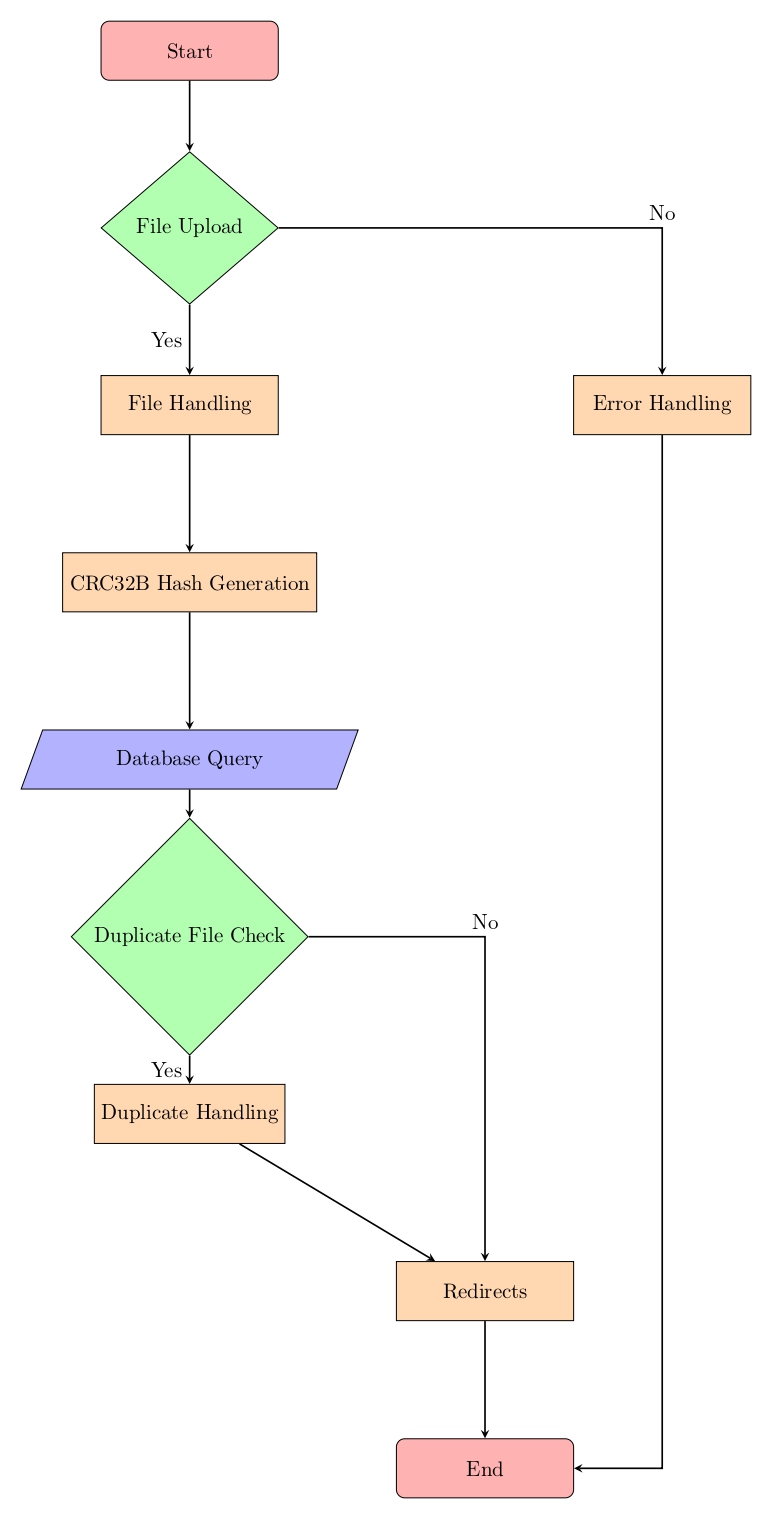
Four crucial characteristics that are essential to the effectiveness of hashing algorithms were assessed throughout the review process:

* **Security:** Security measures how resistant the algorithm is to collisions, assaults, and weaknesses. Data integrity and confidentiality are supported by algorithms with high levels of security.
* **Accuracy:** Accuracy refers to an algorithm's capacity to produce different hash results for various input data. Even with slight input changes, high accuracy algorithms generate hash values that drastically differ.
* **Authenticity:** Authenticity refers to an algorithm's capacity to validate the accuracy of data and the veracity of its sources. Any tampering or illegal changes to the input data can be successfully detected by algorithms with strong authenticity features (Bellare, 2006).
* **Efficiency:** Efficiency is concerned with the computing speed and resource needs of the algorithm. Efficiency is a measure of an algorithm's ability to balance quick hash creation with little computational overhead.

Here, we give a detailed explanation of the experimental setup and attribute grading system that were used to assess each hashing algorithm. As part of the evaluation process, attribute ratings for security, correctness, authenticity, and efficiency are calculated on a scale from 1 to 5.

* **Security Attribute Rating:** We carried out several tests and analysis to determine the security attribute rating for every hashing method. The security evaluation covered a range of topics related to cryptographic robustness, attack resistance, and breach susceptibility. The standards used to determine security ratings were carefully developed, taking into consideration industry norms and recommended procedures. A thorough presentation of the technique used provides insight into the reasoning behind the scores that were assigned.
* **Accuracy Attribute Rating:** Each hashing algorithm's correctness was carefully assessed in order to determine how well it could produce separate hash values for various data inputs. We evaluated the algorithm's capacity to distinguish data with minute differences through tests and data analysis. The accuracy score calculation algorithm and methods are explained, guaranteeing the evaluation process's reproducibility.
* **Authenticity Attribute Rating: The** algorithm's efficacy in confirming the veracity and correctness of data is examined in detail by the authenticity attribute rating. To replicate situations when data authenticity and integrity are crucial, experimental protocols were created. The steps involved in determining authenticity ratings are described, offering valuable information on the validity of the evaluation procedure.
* **Efficiency Attribute Rating:** A series of tests and quantitative analysis were used to evaluate efficiency, a crucial quality. We assessed how well the algorithm managed data by taking into account variables including processing speed, resource use, and data management optimization. A thorough comprehension of the efficiency evaluation is ensured by the precise formula and methodology used to give efficiency ratings.

1. **Evaluation Criteria**



To evaluate each algorithm's characteristics statistically, a score system was developed. Algorithms were given ratings based on how well they performed in each attribute on a scale from 1 to 5. The following evaluation criteria were used:

Security Analysis: In terms of cryptographic resilience, the security attribute evaluation revealed the advantages and disadvantages of each method. Higher ratings were achieved by cryptographic hash algorithms like SHA-1 and SHA-384 because of their demonstrated resilience to cryptographic assaults. In contrast, checksum generation methods like Adler-32 and CRC32b, which are not mainly intended for cryptographic applications, had lower security scores.

Accuracy Evaluation: The accuracy study showed how successfully each method generated unique hash values for a range of input data. The precision of cryptographic methods like MD5 and SHA-1 was higher, producing hash values that varied dramatically even when the input changed slightly. Despite being effective, Adler-32 and CRC32b have significantly less accuracy since they focused mostly on checksum production.

Authenticity Analysis: Evaluation of the authenticity attribute gave information on how well each algorithm checked the accuracy and legitimacy of the data. In this regard, cryptographic hash algorithms like SHA-256 excelled, providing reliable procedures for data validation. Although not intended for robust authenticity, Adler-32 and CRC32b nonetheless displayed reasonable ratings, successfully identifying data modifications in some use scenarios.

Efficiency Analysis: The analysis of efficiency provided information on the computational performance and resource requirements of each program. Due to their design for quick hash generation, algorithms like MurmurHash3 and CRC32b have become the leaders in efficiency. Although they established a balance between security and economy, cryptographic algorithms like SHA-1 and SHA-384 had a little greater processing cost.

1. **Algorithms’ Attributes (results)**

CRC32b Algorithm: The CRC32b method, which is renowned for being straightforward and effective, is frequently used for integrity and error checking in a variety of applications. Despite not being intended for cryptographic strength, it is a good fit for situations where data integrity and quick processing are crucial due to its high efficiency and precise mistake detection capabilities (M. A. Ayub, 2019).

|  |  |
| --- | --- |
| Attribute | Score |
| Security | 3 |
| Accuracy | 4 |
| Authenticity | 3 |
| Efficiency | 5 |

**Graph With Table 1: Algorithm: CRC32b**

MD5 Algorithm: The MD5 algorithm produces fixed-length hash results that are still useful for applications like checksum verification. It was previously a commonly used cryptographic hash function. Despite its historical importance, MD5's relevance in secure applications has been reduced by its susceptibility to collision attacks. Nevertheless, it remains useful in some non-cryptographic use cases due to its quick hash generation and wide compatibility.

|  |  |
| --- | --- |
| Attribute | Score |
| Security | 4 |
| Accuracy | 5 |
| Authenticity | 4 |
| Efficiency | 3 |

**Graph With Table 2: Algorithm: MD5**

SHA-1 Algorithm: SHA-1 is a member of the Secure Hash Algorithm family and has become well-known for its security features. But over time, flaws appeared that diminished its ability to withstand collisions (Nitschke, 2017). Although it is no longer advised for secure applications, legacy systems and some non-critical use cases nevertheless make use of it due to its pervasive integration and compatibility.

|  |  |
| --- | --- |
| Attribute | Score |
| Security | 4 |
| Accuracy | 4 |
| Authenticity | 4 |
| Efficiency | 3 |

**Graph With Table 3: Algorithm: SHA-1**

SHA-384 Algorithm: The SHA-384 algorithm, a member of the SHA-2 family, provides more security than its forerunners. Its resistance against cryptanalytic assaults is strengthened by its greater hash size and complicated procedures. The security features of SHA-384 meet modern security requirements with applications ranging from digital signatures to data verification.

|  |  |
| --- | --- |
| Attribute | Score |
| Security | 5 |
| Accuracy | 4 |
| Authenticity | 4 |
| Efficiency | 2 |

**Graph With Table 4: Algorithm: SHA-384**

GOST Algorithm: The GOST algorithm, which emerged from the Soviet Union during the Cold War, is a notable participant in the hashing industry. It distinguishes itself from Western algorithms via its distinctive design and historical relevance. Although it doesn't have the same level of cryptographic security as some of its competitors, the way it works and the applications it is used in in some places demonstrate how relevant it is still today.

|  |  |
| --- | --- |
| Attribute | Score |
| Security | 3 |
| Accuracy | 3 |
| Authenticity | 3 |
| Efficiency | 3 |

**Graph With Table 5: Algorithm: GOST**

MD2 Algorithm: The MD2 hashing method, created by Ronald Rivest, served as a foundation for later research in cryptography. Its accessibility for instructional purposes was facilitated by its simplicity, but its usefulness in contemporary secure systems is constrained by its flaws and vulnerability to collision attacks.

|  |  |
| --- | --- |
| Attribute | Score |
| Security | 2 |
| Accuracy | 2 |
| Authenticity | 2 |
| Efficiency | 4 |

**Graph With Table 6: Algorithm: MD2**

Adler-32 Algorithm: Adler-32 focuses on creating checksums for data integrity verification, as opposed to cryptographic hash functions. It is appropriate for situations where real-time validation is required due to its effectiveness and quick mistake identification. Although not designed for cryptographic applications, its lightweight nature makes it appropriate for usage in situations where efficiency is crucial.

|  |  |
| --- | --- |
| Attribute | Score |
| Security | 2 |
| Accuracy | 3 |
| Authenticity | 3 |
| Efficiency | 5 |

**Graph With Table 7: Algorithm: Adler-32**

MurmurHash3 Algorithm: MurmurHash3 excels as a fast, high-quality algorithm. Its efficiency, attained through a simplified methodology, supports contexts where performance is crucial. Although MurmurHash3 is not intended for use in high-security cryptographic applications, its capacity to generate trustworthy hash values quickly makes it a useful tool in situations where computing speed is essential.

|  |  |
| --- | --- |
| Attribute | Score |
| Security | 3 |
| Accuracy | 4 |
| Authenticity | 3 |
| Efficiency | 5 |

**Graph With Table 8: MurmurHash3**