Programme Name and Semester: Masters of Computer Application, 4th Semester
Course Name: Introduction to Blockchain Technology and Application
Course Code: MCA403B
Academic Session: 2023-2024 Even

Study Material_3

(Introduction to Blockchain Technology and Application [MCA403B])

Module 3: Basic Crypto primitives [06H]

- Hash functions
- Puzzle friendly Hash
- Collison resistant hash
- Digital signatures
- Public key crypto
- Verifiable random functions
- Zero-knowledge systems.

● Hash functions;

An integral component of computer science applications, hash functions serve as mathematical algorithms that convert input, often referred to as a 'message', into fixed-size strings of bytes. Their significance spans across diverse fields, including data retrieval, cryptography, and data integrity verification. In these applications, hash functions play pivotal roles, enabling efficient organization and manipulation of data while ensuring security and integrity in various computational processes.

Functionality and Purpose:

- Hash functions excel in mapping data of arbitrary sizes to fixed-size values.
- While most hash functions produce fixed-size outputs, some support variable-length outputs.
- The resulting values from a hash function are termed hash values, codes, digests, or simply hashes.
- These hashes are commonly employed to index hash tables, facilitating efficient data storage and retrieval.

Utilization in Data Storage:

- Hash functions, in conjunction with hash tables, play a crucial role in data storage and retrieval systems.
- Retrieving data from a hash table typically involves a small and nearly constant time per retrieval, offering efficiency benefits.
- The storage space required for hash tables is only marginally greater than that needed for the data or records themselves.

Statistical Properties and Performance:

- The efficacy of hash functions relies on statistical properties governing the interaction between keys and the function itself.

- While worst-case scenarios may exhibit intolerably poor performance, they are typically rare occurrences.
- On average, hash functions demonstrate nearly optimal behaviour, minimizing collision occurrences and enhancing efficiency in data retrieval tasks.

Efficiency of Hash Functions: Hash functions offer several advantages across various fields of computer science and cryptography:

1. **Data Integrity Verification:** Hash functions are commonly used to verify the integrity of data by generating a unique hash value for a given dataset. Any alterations to the data, even minor changes, will result in a different hash value, allowing for easy detection of tampering or corruption.
2. **Data Retrieval Efficiency:** Hash functions are utilized in data storage and retrieval systems, particularly with hash tables. They enable efficient indexing and retrieval of data, often providing nearly constant time access, which can significantly improve performance compared to other data structures like lists or trees.
3. **Cryptography:** Hash functions play a crucial role in cryptographic applications, such as digital signatures, message authentication codes (MACs), and password hashing. They provide one-way functions that make it computationally infeasible to reverse-engineer the original input from its hash value, ensuring data security and confidentiality.
4. **Uniqueness and Identifiers:** Hash functions produce unique hash values for different inputs, minimizing the likelihood of hash collisions (where two different inputs produce the same hash value). This uniqueness property is valuable in generating identifiers, such as file checksums or unique identifiers in distributed systems.
5. **Consistency and Reproducibility:** Hash functions ensure that the same input always produces the same hash value, regardless of when or where the calculation occurs. This consistency and reproducibility are essential for various applications, including caching mechanisms and data synchronization.
6. **Space Efficiency:** Hash functions typically generate fixed-size output, regardless of the input size. This fixed-size representation allows for efficient use of storage space, making hash functions suitable for applications with limited memory or storage resources.
7. **Fast Computation:** Many hash functions are designed to be computationally efficient, enabling rapid generation of hash values even for large datasets. This speed is advantageous in real-time systems, where quick data processing is essential.
8. **Versatility:** Hash functions are versatile and can be adapted to various requirements and constraints. Different hash functions are available with varying properties, such as cryptographic strength, collision resistance, and performance characteristics, allowing users to choose the most suitable one for their specific needs.

- **Limitation of Hash Function.** While hash functions offer numerous benefits and are widely used in various applications, they also have some limitations:

1. **Collision Vulnerability:** One of the primary limitations of hash functions is the possibility of collisions, where two different inputs produce the same hash value. While modern cryptographic hash functions aim to minimize collision probabilities, they are not entirely collision-resistant. Collisions can potentially be exploited to undermine the security of systems relying on hash functions.
2. **Limited Input Space:** Hash functions typically produce fixed-size output regardless of the input size. This can lead to information loss, especially when hashing large datasets or variable-length

inputs. In some cases, different inputs may produce the same hash value due to this fixed-size constraint, increasing the likelihood of collisions.

3. **Deterministic Nature:** Hash functions are deterministic, meaning the same input always generates the same hash value. While this property is essential for consistency and reproducibility, it also means that hash functions lack variability, making them vulnerable to pre-image attacks. Attackers can potentially reverse-engineer the original input from its hash value using brute-force or dictionary attacks.
4. **Performance Overhead:** While many hash functions are designed for efficiency, hashing large datasets or complex inputs can incur performance overhead. The computational complexity of hash functions may impact system performance, especially in real-time or resource-constrained environments. Additionally, cryptographic hash functions, which prioritize security, may be computationally intensive.
5. **Hash Flooding Attacks:** Hash functions used in hash tables and similar data structures are susceptible to hash flooding attacks. Attackers can deliberately craft input data to generate a large number of collisions, causing hash table collisions and degrading performance. Mitigating hash flooding attacks typically requires implementing collision-resistant hash functions or employing techniques like chaining or probing.
6. **Cryptographic Weaknesses:** Some hash functions, particularly older or less secure algorithms, may exhibit cryptographic weaknesses over time. As computing power advances and new attack techniques emerge, hash functions previously considered secure may become vulnerable to exploitation. It's essential to use cryptographically secure hash functions and regularly update cryptographic protocols to mitigate such risks.
7. **Hash Function Design Vulnerabilities:** Poorly designed hash functions may exhibit structural vulnerabilities or algorithmic flaws that can be exploited by attackers. Weaknesses in hash function design can lead to security vulnerabilities, including pre-image attacks, collision attacks, and length extension attacks. It's crucial to use hash functions that have undergone rigorous cryptographic analysis and testing to minimize these risks.

   Despite these limitations, hash functions remain indispensable tools in modern computing, cryptography, and data management systems. By understanding these limitations and adopting appropriate mitigation strategies, users can leverage hash functions effectively while minimizing security risks and performance overhead.

● <u>Functioning of Hash Functions:</u>

A hash function takes a key as an input, which is associated with a datum or record and used to identify it to the data storage and retrieval application. The keys may be fixed length, like an integer, or variable length, like a name. In some cases, the key is the datum itself. The output is a hash code used to index a hash table holding the data or records, or pointers to them. A hash function may be considered to perform three functions:

- Convert variable-length keys into fixed length (usually machine word length or less) values, by folding them by words or other units using a parity-preserving operator like ADD or XOR.
- Scramble the bits of the key so that the resulting values are uniformly distributed over the key-space.
- Map the key values into ones less than or equal to the size of the table

Algorithm:

- Simple Hash(input)
- Input: input (string or data)
- Output: hash (hash value)

```
1. Initialize hash value as 0 or any predetermined constant.
2. For each character in the input:
   a. Convert the character to its ASCII value.
   b. Update the hash value by adding the ASCII value of the character.
     ● Return the hash value.
```

Application of Hash Function.

Hash functions find applications across various domains due to their versatility, efficiency, and security properties. Some common applications include:

1. **Data Integrity Verification:** Hash functions are used to ensure the integrity of data during transmission or storage. By generating a hash value for the original data and comparing it to the hash value of the received or stored data, one can quickly detect any alterations, corruption, or tampering.
2. **Cryptographic Applications:** Hash functions serve as fundamental building blocks in modern cryptography. They are utilized in digital signatures, message authentication codes (MACs), password hashing, and key derivation functions (KDFs). Hash functions play a crucial role in ensuring data confidentiality, integrity, and authenticity in secure communication protocols and systems.
3. **Data Retrieval and Storage:** Hash functions are employed in data structures such as hash tables, which enable efficient storage and retrieval of data. Hash tables offer constant-time complexity for basic operations like insertion, deletion, and lookup, making them suitable for applications requiring fast data access, such as databases and caches.
4. **Content Addressable Storage (CAS):** In content addressable storage systems, hash functions are used to generate unique identifiers or keys for data. This allows for quick and efficient storage and retrieval of content based on its hash value, eliminating the need for traditional file paths or directory structures.
5. **Distributed Systems and Peer-to-Peer Networks:** Hash functions play a crucial role in distributed systems and peer-to-peer networks for data partitioning, load balancing, and routing. Hash values are often used to determine the location or responsibility of data replicas, peers, or nodes in the network, facilitating efficient data distribution and communication.
6. **Checksums and Error Detection:** Hash functions are utilized to generate checksums for verifying the integrity of files or data blocks. By comparing the checksum of the original data with the calculated checksum, errors or discrepancies can be detected, enabling reliable error detection and correction mechanisms in data transmission and storage systems.
7. **Blockchain and Cryptocurrencies:** Hash functions are integral to the functioning of blockchain technology and cryptocurrencies like Bitcoin. Hash functions are used to create cryptographic hash pointers, which link blocks of transactions together, ensuring immutability, transparency, and security in decentralized ledgers.

8. **Digital Forensics and Data Deduplication:** Hash functions are employed in digital forensics to uniquely identify files and digital artefacts. By comparing hash values of files, investigators can identify duplicates, detect unauthorized modifications, and trace the provenance of digital evidence. Hash functions also play a crucial role in data deduplication, where redundant data is identified and eliminated based on their hash values, reducing storage space and improving efficiency.

These are just a few examples of how hash functions are applied in various fields. Their versatility and effectiveness make them indispensable tools in modern computing and information security.

Puzzle friendly Hash

A puzzle-friendly hash, in the context of cryptography and computer science, typically refers to a hash function that is designed to be computationally expensive and time-consuming to evaluate. This property makes it more resistant to brute-force attacks and adds a layer of security to the hashing process. Puzzle-friendly hashes are often used in proof-of-work systems and other cryptographic applications.

The idea is to make it difficult for an attacker to compute the hash value by requiring significant computational effort. This effort could be achieved through techniques like key stretching, where the hash function is applied multiple times in a loop, or by using functions that involve complex mathematical operations. An example of a puzzle-friendly hash function is Scrypt, which is designed to be memory-hard and computationally intensive. It is commonly used in cryptocurrencies like Litecoin to make mining more challenging and resistant to specialized hardware attacks.

It's important to note that puzzle-friendly hashes are primarily used in specific scenarios, such as proof-of-work systems, and may not be suitable for all applications. In many cases, a regular cryptographic hash function like SHA-256 or SHA-3 is sufficient for general-purpose use.

Definition:

Here is the formal technical definition of the puzzle friendliness property.[2][1]

- A hash function H is said to be *puzzle friendly* if for every possible n-bit output value y, if k is chosen with a distribution with high min-entropy, then it is infeasible to find x such that $H(k \| x) = y$ (where the symbol "||" denotes concatenation) in time significantly less than 2^n.

In the above definition, the distribution has high min-entropy means that the distribution from which k is chosen is hugely distributed so that choosing some particular random value from the distribution has only a negligible probability.

Why this property is called puzzle friendliness?

Let H be a cryptographic hash function and let an output y be given. Let it be required to find z such that H (z) $= y$. Let us also assume that a part of the string z, say k, is known. Then, the problem of determining z boils down to finding x that should be concatenated with k to get z. The problem of determining x can be thought of a puzzle. It is really a puzzle only if the task of finding x is nontrivial and is nearly infeasible. Thus, the puzzle friendliness property of a cryptographic hash function makes the problem of finding x closer to being a real puzzle.

Application in cryptocurrency:

The puzzle friendliness property of cryptographic hash functions is used in Bitcoin mining.

A puzzle-friendly hash, in the context of cryptography and computer science, typically refers to a hash function that is designed to be computationally expensive and time-consuming to evaluate. This property makes it more resistant to brute-force attacks and adds a layer of security to the hashing process. Puzzle-friendly hashes are often used in proof-of-work systems and other cryptographic applications.

The idea is to make it difficult for an attacker to compute the hash value by requiring significant computational effort. This effort could be achieved through techniques like key stretching, where the hash function is applied multiple times in a loop, or by using functions that involve complex mathematical operations.

An example of a puzzle-friendly hash function is Scrypt, which is designed to be memory-hard and computationally intensive. It is commonly used in cryptocurrencies like Litecoin to make mining more challenging and resistant to specialized hardware attacks. It's important to note that puzzle-friendly hashes are primarily used in specific scenarios, such as proof-of-work systems, and may not be suitable for all applications. In many cases, a regular cryptographic hash function like SHA-256 or SHA-3 is sufficient for general-purpose use.