Blockchain CheatSheet - Hashing

Table of Contents

---

§ Fundamentals

Hashing is not encryption because you cannot rebuild the original data from the hash as you can with encrypted files.

We should consider hashing like a fingerprint; it provides a secure genetic reference to the data but is not the data "in person".

Key Characteristics of Good Cryptographic Hashing

1. **Speed**: It must be easy to compute to a certain extent because we don't want the algorithm to be easily brute-forced due to its speed.
2. **Deterministic**: The same input should always produce the same output.
3. **One-way**: It must be infeasible to recreate the original data from the hash. In particular it is hard because during hashing we may lose data.
4. **Secure**: If you alter the data to be hashed, you get a totally different hash, but if you re-alter back, you get the original hash.
5. **Collision**: It is impossible for two different data sets to have the same hash value, so hashing is Concurrency-safe *.
6. **Size**: It doesn't matter how large is the data to be hashed. Since the Hashing practice has generally a lot of capacity.

> * **Collision Issue**: The primary concern is not just the probability of any two hashes colliding, but rather the likelihood that, within a dataset, there will be at least two identical data points with the same hash. This probability increases significantly with the size of the dataset, similar to the birthday paradox.

Salting

Salting is the practice of adding a random value to the hashed password stored. This is the only way to securely hash passwords.

Miners

The task of miners is to take transactions or data from the blockchain buffer and group them into blocks. Each block header is 80 bytes in size.

Before adding these blocks to the blockchain, miners must include a 32-byte hash and a nonce that meets the current difficulty requirement.

They do this by cycling through different nonce values until they find one that produces a hash satisfying the proof-of-work condition.

---

§ Hashing Math

Overview

- ***SHA (Secure Hash Algorithm)***:

```
1. SHA-1: 160 bits
2. SHA-2:
        - SHA-224: 224 bits
        - SHA-256: 256 bits
        - SHA-384: 384 bits
        - SHA-512: 512 bits
3. SHA-3:
        - SHA3-224: 224 bits
        - SHA3-256: 256 bits
        - SHA3-384: 384 bits
        - SHA3-512: 512 bits
```

- ***Technical Terms***:
  - **Padding**: Adding bits to indicate the end of the message.
  - **Padding with Zeros**: Adding '0' bits to reach a specific length.
  - **Append Length**: Adding the original length of the message in bits.
  - **Compression Function**: The process of mixing bits that includes cryptographic operations.

- **Hash Value**: The resulting unique secret code.

Procedure

1. Prepare the Message

Our Case: Imagine you have a phrase, for example: "Hello World". This is our input. Calculate the length of the input in bits (88 bits in this case).

2. Add an End Signal (Padding)

To let the algorithm know that the phrase is finished, we add a special symbol at the end.

Our Case: We add a '1' bit. This signal is the padding bit. So now we have "Hello World1".

3. Block Structures

The algorithm prefers to work with blocks of a certain size, as a computing power optimization. For SHA-256, the block size is 512 bits (64 bytes) at a time.

Small Data - Add Missing Pieces (Padding with Zeros)

Our Case: If the phrase is not long enough like "Hello World1", we add zeros to fill it up. So, if "Hello World1" is 88 bits long, we add another 424 zeros to make it 512 bits.

Big Data - Portioning

If data that has to be hashed is longer than 512 bits, the algorithm runs multiple times on chunks.

4. Add the Length (Append Length)

At the end, we append the length of the original message in bits, as required by the SHA-256 padding rules.

Our Case: "Hello World" was 88 bits, so we add a 64-bit representation of "88". Now we have a total of 512 bits: 448 bits of data and padding + 64 bits of length.

5. Mix the Characters (Compression Function)

Now the algorithm starts mixing the characters. It takes each 512-bit block and performs a lot of complex operations on them, changing the bits in a very complicated way that only the algorithm knows. This step includes operations like XOR, bitwise shifts, and modular additions.

Big Data - Merkle Root

The algorithm takes all the 512 bits chunks and concatenates them in pairs executing the hashing again and again until having as a result a 256 bits Hash

Technically executing a Collapsing on long groups of hashed data into a single Hash called The **Merkle Root**

6. Obtain the Secret Code (Hash Value)

After the algorithm has finished mixing, we get a unique secret code called a hash or digest, like "a7b9c3d2". This code is special because even if you change just one letter of the original message, the hash will be completely different.

---

§ Applications

Is useful to check if some data has been corrupted or modified in a defined period of time since creation or to certify the origins of a data. This is possible by checking the Hash from T0 with T1.

Suggested Follow-up

Blockchain CheatSheet - Cryptography & Signatures

Author: Kenneth Boldrini