**The Block Chain**

**Hash Function**

A Hash Function is any function that can be used to map data of arbitrary size to data of fixed size. The values returned after are called: hash values, hash codes, digests or simply hashes.

This can be used with a data structure called a hash table used for rapid data lookup, the hash function accelerates table or database lookup by detecting duplicated records in a large file.

**1.Uses (authentication, integrity)**

Hash tables: Used to quickly locate a data record given a key, it maps the key to a list to lookup for the corresponding record stored. The domain of a hash function (possible set of keys) is larger than its range (number of different table indices) and so will map several different keys to the same index that could result in a collision. So each slot(bucket) of a hash table is associated with a set of records than a single one. Example of hash function: Jenkins hash or Zobrist hashing.

Caches: It is simpler than a hash table, since the collision can be resolved by discarding or writing back the folder collided.

Finding duplicate records: The hash function map each record to an index into a table T, and collect each bucket T[i] a list of numbers of all records with the same hash value i. After, any two duplicate records will end in the same bucket. Then the duplicates can be found by scanning the bucket T[i] containing two or more members and will be compared.

Protecting data: A hash value can be used to identify secret information, so the hash function must be collision-resistant (difficult to find data generating the same hash value). We call these function cryptographic-secure-hash functions. The collision resistance is accomplished by generating large hash values. Example of secure hash function: SHA-1.

Finding similar records: Hash function can be used to locate table records whose key is similar to a given key. A hash function maps similar keys to hash values that differ at most m (1-2). Looking in a table T using this hash function, the similar records will end up in the same or nearby bucket. Then it only checks the records in the bucket T[i] to the T[i+k] with k ranges (-m, m). Example: acoustic fingerprint.

Finding similar substring: The same techniques can be used to find equal or similar stretches in a collection of string such as a document or a database. In this case the strings are broken into pieces and a hash function detect the potentially equal pieces. Example: Rabin-Karp algorithm.

Geometric hashing: Used in communication, computer graphics, geometry, proximity problems in plane or three dimensional space. Used to find closest pairs of points, similar shapes or image. Here the input is a metric space and the hash function a partition of that space. The table is often an array of two or more indices and the hash function returns an index tuple.

**2.Properties**

Determinism: For a given input value it must always generate the same hash value. It must a function of the data to be hashed. This is a context of the reuse of the function.

Uniformity: A hash function should map the expected inputs over its output range. Every hash value in the output range should be generated with the same probability. The cost of hashing-based methods goes up as the number of collisions increases. We can test it with the chi-squared test.

Defined range: It is desirable that the output of a hash function have a fixed size, it can be used to accelerate data searches. But, cryptographic hash functions produce much larger hash values to ensure the computational complexity of brute-force inversion(SHA-1). Producing fixed-length can be accomplished by breaking the input data into chunks of specified size. In cryptographic hash function, these chunks are much bigger than the hash value.

Variable range: The range of hash values may be different for each run. In this situation, the hash function may take two parameters-the input data z and the number n of allowed hash values. A solution to this is to compute a fixed hash function with a large range, divide the result by n and use the division remainder. [READ AGAIN]

Variable range with minimal movement (dynamic hash function): If the hash function store values in a hash table outliving the run of the program, the hash table need to be expanded or shrunk and referred as a dynamic hash table. As example, we have linear hashing (execute in constant time but relax the property of uniformity) or extendible hashing.

Data normalization: Some input data may contain features irrelevant for comparison purposes. For those data, the hash function may be compatible with the data equivalence criterion being used, so we normalize the input.

Continuity: A hash function used to search similar data must be continuous. Two inputs differing a little should be mapped to equal or nearly equal hash values. The continuity wouldn’t be used for cryptographic hash function.

Non-invertible: In cryptographic, hash functions are expected to be non-invertible (can’t find back an input x from its hash value h(x)).

**3.Hash function algorithms**

Trivial hash function: If the data to be hashed is small enough, we can use the data itself as the hashed value, so there is zero cost computing. This hash function is called perfect as it maps each input to a distinct hash value.

Perfect hashing: An injective hash function (map each valid input to a different hash value) is said to be perfect.

Minimal perfect hashing: A perfect hash function is said to be minimal if the range of input is equal to the range of the table. In others words, it yields a compact hash table without any vacant slots.

Hashing uniformly distributed data: If the inputs are strings, and may independently occur with uniform probability, then the hash function needs to map the same number of inputs to each hash value.

Hashing variable-length data: If the data values are long character strings, the hash function needs to depends on all the characters of the string, and depends on each character in a different way.