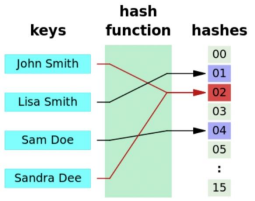
DMC001 Hash Contract: Creating Contracts using Hash Functions

Liang Ng

November 2023

The definition of hash function in Wikipedia, while technically correct, is (perhaps unintentionally) misleading (obviously to non-experts) as it singles out hash tables as its primary application. We shall revisit and review this definition after exploring some other novel applications in this article.

A **hash function** is any function that can be used to map data of arbitrary size to fixed-size values, though there are some hash functions that support variable length output.[1] The values returned by a hash function are called *hash values*, *hash codes*, *digests*, or simply *hashes*. The values are usually used to index a fixed-size table called a *hash table*. Use of a hash function to index a hash table is called *hashing* or *scatter storage addressing*.



A hash function that maps names to integers from 0 to 15. There is a collision between keys "John Smith" and "Sandra Dee".

Hash functions and their associated hash tables are used in data storage and retrieval applications to access data in a small and nearly constant time per retrieval. They require an amount of storage space only fractionally greater than the total space required for the data or records themselves. Hashing is a computationally and storage space-efficient form of data access that avoids the non-constant access time of ordered and unordered lists and structured trees, and the often

(We shall skip comparisons with Blockchain Smart Contract in this article to save time for readers – the most precious commodity in TikTok age.)

- A hash function is any function that can be used to map data of arbitrary size to fixed-size values.

One of the most useful novel applications of hash function is to convert message text of arbitrary length into a fixed length integer, the shortest practical size being 53 bits, formed using JavaScript IEEE 64 bits floating point number, by appending 52 bit mantissa with the sign bit. Technically competent readers will be aware of longer hash codes. This article shall focus on examples using 53 bit hashes. Readers may apply longer hash codes to ideas discussed here easily.

DMeta Hash Contract demo page below demonstrates the applications of 53 bit hash code:

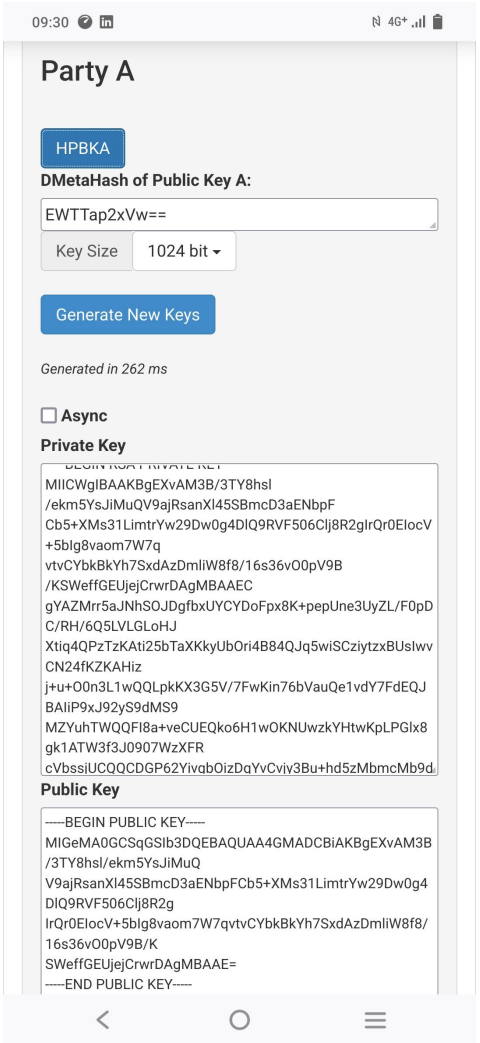
- https://godmeta.github.io/dmcontract/

The following contract text is converted into DMeta Hash (53 bit hash in base 64 code) when the button DMetaHash is pressed:

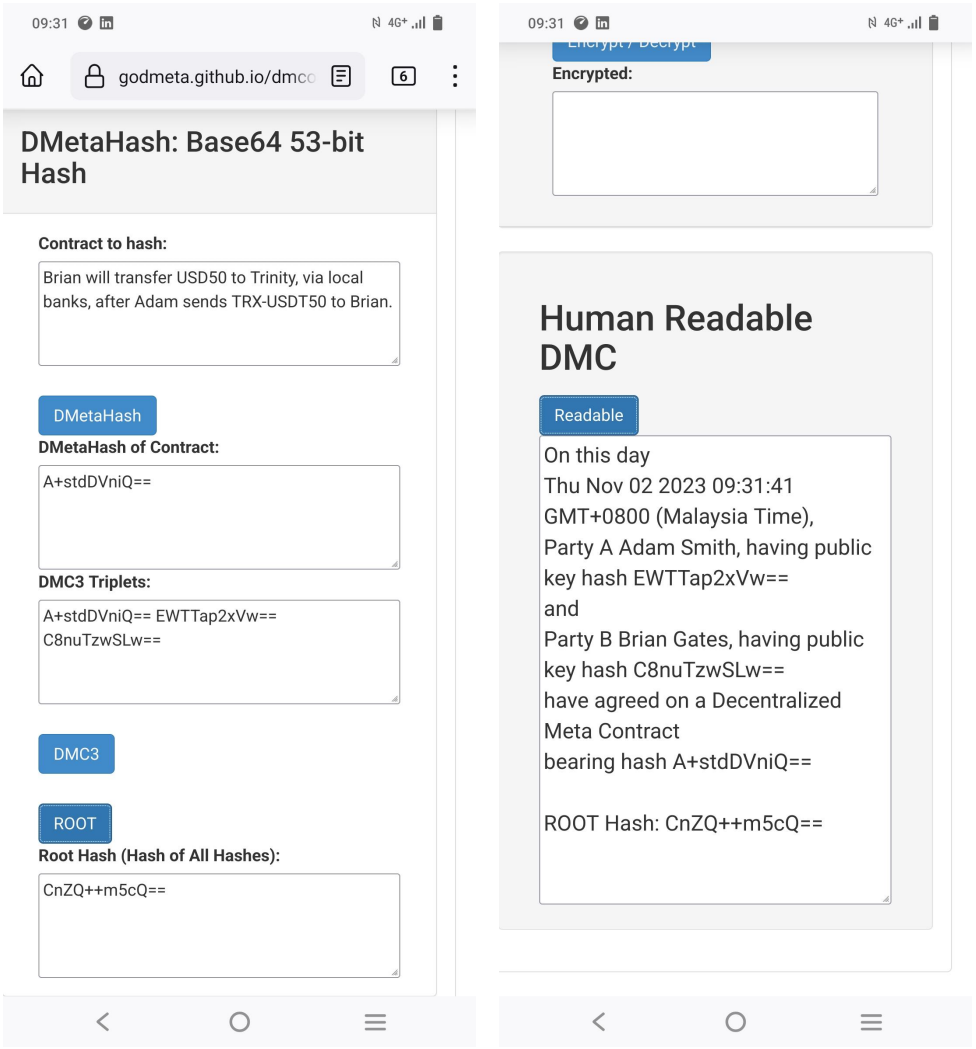
- Brian will transfer USD50 to Trinity, via local banks, after Adam sends TRX-USDT50 to Brian.

The corresponding DMeta Hash is A+stdDVniQ== (known as Contract Hash, as it is derived from the Contract text above.)

In addition, two other hashes HPBKA (Hash of Public Key of Party A) and HPBKB (Hash of Public Key of Party B) are generated, by pressing corresponding buttons, as shown in screenshots below.



Next, the three hashes are concatenated by pressing DMC3 button (DMeta Contract Triplets).



Finally, ROOT button is pressed to generate the Root Hash, which is a hash of DMeta Contract Triplets.

Concatenative Properties

Hash codes have concatenative properties: they can be concatenated in two ways:

1. Spatially: appended to one another, with delimiter.
2. Sequentially: the output of a hash function which is a string can be input to the same hash function, and so on.

As such, the Root Hash or H0 is an integer (53 bit, coded as base 64 string) that is cryptographically unique (within collision limits) representing the Contract between Party A and Party B (hence known as DMeta Contract).