Algorithms & Techniques

Two techniques are predominantly used for securing the chain and for efficient validation and verification:

Hashing and asymmetric key encryption.

We'll examine just one common use of asymmetric key encryption.

Let's say a participant in Buffalo wants to transact with the participant in Kathmandu. Instead of sending just a simple message, a participant in Buffalo will send a transaction data encrypted by Buffalo's private key, and then encrypted by Kathmandu's public key. Kathmandu will first decrypt the data using its own private key, then use Buffalo's public key to decrypt assigned transaction data. This ensures that only Kathmandu can decrypt and receive the data and that only Buffalo could have sent the data.

A popular implementation of public key, private key is the Rivest Shamir Adleman (RSA) algorithm.

Though RSA is very commonly used in many applications, block chains need a more efficient and stronger algorithm.

Elliptic Curve Cryptography, ECC family of algorithms is used in the bitcoin as well as an Ethereum block chain for generating the key pair.

ECC is stronger than RSA for a given number of bits.

256-bit ECC key pair is equal in strength to about 3072 bits of RSA key pair.

Both bitcoin and Ethereum use ECC based algorithms for their encryption needs.

Hashing:

A hash function or hashing transforms and maps an arbitrary length of input data value to a unique fixed length value.

Input data can be a document, tree data, or a block data. Even a slight difference in the input data would produce a totally different hash output value.

Hashing Requirements:

- Algorithm should be one-way function
  The first requirement is to make certain that no one can derive the original items hashed from the hash value.

- Collision Free
  The second requirement is to make sure that the hash value uniquely represents the original items hashed.

These requirements are achieved by choosing a strong algorithm such as secure hash, and by using appropriately large number of bits in the hash value.

Most common hash size now is 256 bits and the common algorithms are **SHA-3**, **SHA-256** and **Keccak.**

Two different approaches for Hashing:

- A simple Hash
  In the simple hash approach, all the data items are linearly arranged and hashed.

- A Merkle tree hash
  In a tree-structured approach, the data is at the leaf nodes of the tree, leaves are pairwise hash to arrive at the same hash value as a simple hash.

When to use simple hash?

Fixed number of items to be hashed (Ex- Block Header)

Verifying composite block integrity

When to use tree hash?

When number of items differ from block to block

In Ethereum, hashing is used to generate:

- Account Addresses
- Digital Signatures
- Transaction Hash
- State Hash
- Receipt Hash
- Block Header Hash

Transaction Integrity:

To manage the integrity of a transaction we need:

- Secure a unique account address
- Authorizing of the transaction by the sender through digital signing
- Verification of the content of the transaction is not modified

We use a combination of hashing and public key cryptography to solve these problems.

Addresses of accounts are generated using public key, private key pair.

- Step 1, at 256-bit random number is generated, and designated as the private key.
- Step 2, an ECC algorithm is applied to the private key, to get a unique public key. This is the private public key pair.
- Step 3. Then a hashing function is applied to the public key to obtain account address.

The address is shorter in size, only 20 bytes or 160 bits.

Data is hashed and encrypted. This is the digital signature. The receiver gets the original data, and the secure hash digitally signed.

Receiver can recompute the hash of the original data received, and compare it with the received hash to verify the integrity of the document.

Now, consider the transaction to be that data:

- Step number 1, find the hash of the data fields of the transaction.
- Step number 2, encrypt that hash using the private key of the participant originating the transaction. Thus, digitally signing the transaction to authorize and making the transaction non-repudiable.
- Step number 3, this hash is added to the transaction. It can be verified by others decrypting it using the public key of the sender of the transaction, and recomputing the hash of the transaction. Then, compare the computed hash, and the hash received at the digital signature. If there is a match, accept the transaction. Otherwise, reject it.

A combination of hashing and encryption are used for securing the various elements of the block chain.

Private public key pair and hashing are important foundational concepts in decentralized networks that operate beyond trust boundaries.