Blockchains for the Working Mathematician

Elements of Post-Bitcoin Stateful-P2P Networks

`wires`

May 24, 2017

Abstract

1 Preliminaries

We introduce some notation.

1.1 Binary Data

A bit is either zero (false) or one (true)

$\underline{\textbf{bit}}$ `:=` $\{0, 1\}$

A sequence of n bits of is written as

$\underline{\textbf{bits}}$ `:=` $\{0, 1\}^n$

A sequence of bits of arbitrary length is notated as

$\underline{\textbf{bitstring}}$ `:=` $\{0, 1\}^\star$

1.2 Hash Functions

Pick some cryptographically secure hash function H (e.g. `SHA256` or something better)

$$H : \{0, 1\}^\star \to \{0, 1\}^m$$

Such function sometimes called *one way functions* because the inverse H^{-1} or pre-image function $\{x|H(x) = y\}$ cannot be efficiently computed. given just the output y the only way to find a x such that $H(x) = y$ is by brute force, try all possible x until a match is found. (NB. This fact is exploited in Proof of Work, explained later 1.4)

Furthermore, flipping just a single bit of an input string p to yield p', $H(p)$ and $H(p')$ should give radically different addresses. The output of H should be indistinguishable from a pseudo-random function and is should be distributed evenly over **addr** .

1.3 Content Addressing

We fix some size $A = 160$ (for example) and define our *"address space"* whose elements are all possible A-bit strings.

$$\underline{\textbf{addr}} := \{0,1\}^A = \{0,1\}^{160}$$

Idea: We can associate a canonical address in **addr** to every possible **bitstring** by appying H to it.

The size parameter A should be picked large enough so that the hash collision is far more unlikely that the earth being destroyed in the next 1000 years.

$$P(p \neq q | H(p) = H(q)) \ll 1$$

Recall that m is the output size in bits of H. Since typically $m > A$ we just truncate the output of H, taking only the first A bits.

If $m < A$ we should pick a better hash function or make the space smaller (it is possible to use a family of hash functions to construct H of arbitrary large m-parameter).

1.4 Proof of Work

PoW, pioneered by Bitcoin, uses the fact the H is not efficiently invertible to provide a mechanism of proof that one has done a certain amount of computational work.

We introduce a parameter called *difficulty* which is a number $0 < k < m$.

Let $A; B$ denote concatenation of bitstrings.

For a given bitstring X, a *proof of work at difficulty* k, is a bitstring (or number) N called a *nonce*, such that $H(X; N)$ is a bitstring starting with k `0`-bits.

The reasoning is that the only way to find such N is to simply try them then all until one finds that $H(X; N)$ starts with k zeros.

If $k = 1$ there is a 50 percent chance the first bit is zero of every possible nonce. When $k = 2$ it is is 25 percent chance the first two bits are zero, etc.

1.5 Public-Private keys

1.6 Hierarchical Deterministic Wallets

Aka **BIP32**, `bip32.org`

1.7 Mnemonics

Aka **BIP39** aka Brain Wallets

1.8 RLP

Runlength prefix encoding