Blockchain Databases for the Working Mathematician

Ingredients of Stateful Decentralized Peer-to-Peer Networks

`wires`

June 22, 2017

Abstract

Introduction

Research in spam-prevention let to a mechanism called "proof of work". Such proofs are *easy to verify* (algorithmically) but *hard, yet feasible, to construct.*

A "transaction log" is a way to record alterations to a database.

Bitcoin is a system that combines such a transaction log with proof of work to build a fully decentralised database.

The log is used to construct the state of the database and it can only be modified by recomputing the Proof of Work.

> *"It seemed so obvious to me; here we are faced with the problems of loss of privacy, creeping computerization, massive databases, more centralization and Chaum offers a completely different direction to go in, one which puts power into the hands of individuals rather than governments and corporations. The computer can be used as a tool to liberate and protect people, rather than to control them."*
>
> —Hal Finney, Cypherpunks Mailing List, 1992

1 Preliminaries

We introduce various concepts and notations

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

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

A sequence of bits of arbitrary length is notated as

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

1.2 Secure Hashing

The most basic block-chain functionally is provided by cryptographically hash functions.

Hash Function A *hash function* H is any function that can map arbitrary large bit-strings to a fixed size (m) bit string.

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

The result of applying H to some value x is called the *digest* or *the hash value* (of x).

Cryptographically Secure Hash Function A *one way function* is a functions f whose inverse f^{-1} or pre-image $\{x|f(x) = y\}$ cannot be *efficiently computed.*

The existence of one way functions is still an open problem, a proof of which would imply that $P \neq NP$.

Assuming they exist and given just the output y, the only way to find a x such that $f(x) = y$ is by brute force: try all possible values of x until a match is found. This is exploited in Proof of Work, which we talk about later, see 1.5.

A *cryptographically secure hash function* is a hash function that is also a *one way function.*

The output of such hash functions must be indistinguishable from a (pseudo-)random function and the domain should distribute evenly over the co-domain: flipping just a single bit of an input string p to yield p' should have $f(p)$ and $f(p')$ return radically different digest values.

If $p \neq p'$ but $f(p) = f(p')$, we speak of a *hash collision*, this is a *Very Bad Thing.* A hash function is called *collision free* if any efficient algorithm has negligible probability of finding such a collision.

The `SHA` family of hash functions is very well know, `SHA256` is a commonly used cryptographic hash function, considered to be cryptographically secure and collision free.

1.3 Content Addressing

We fix some cryptographically secure hash function $H : X \to \{0,1\}^m$.

The we also fix some *address space* by picking a "size-parameter" A and define our the address space whose elements are all possible A-bit strings.

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

Idea: To every possible **bitstring** x we can now associate a *canonical address* $\tilde{x} \in \mathbf{addr}$ by applying H and truncating the result it to the right size.

$$\tilde{x} = H(x) \upharpoonright \mathbf{addr}$$

We will often need the address of *any* object x, not just bit strings, so we use $\tilde{x}$ as a notation $\tilde{x'}$ where x' is the binary encoding of x.

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).

In any case, when picking A and H should beware that the result remains collision free.

1.4 Asymmetric Cryptography

Cryptography is used to sign and verify transactions, but it can also be used to encrypt and decrypt information. Asymmetric refers to the fact the two operations use different, but related keys, called the *secret* and *public* keys.

Two well know cryptographic systems are *RSA* (Rivest-Shamir-Adleman) and *ECC* (Elliptic Curve Cryptography), however, their details are irrelevant for our purpose.

Let α be some "actor" in our system.

We then SK_α for his or her or it's *secret key*, which you can think of as a very large random number (actually a element of a large finite field).

There is a procedure to derive from SK_α a corresponding *public key*, PK_α.

The two keys together are called a keypair and written as $\langle SK_\alpha, PK_\alpha \rangle$.

We define the actor-address $\tilde{\alpha} := H(PK_\alpha)$.

We can think of the system as having four operations:

- `encrypt`: $\star \times SK_\alpha$

1.5 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.

1.6 Event Sourcing

1.7 State Channels

1.8 Hierarchical Deterministic Wallets

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

1.9 Mnemonics

Aka **BIP39** aka Brain Wallets

1.10 RLP

Runlength prefix encoding

references

```
Dwork, C.; Naor, M. (1993)
Pricing via Processing, Or, Combatting Junk Mail, Advances in Cryptology
CRYPTO92: Lecture Notes in Computer Science No. 740. Springer: 139147
http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.ps

From: Satoshi Nakamoto <satoshi <at> vistomail.com>
Subject: Bitcoin P2P e-cash paper
Newsgroups: gmane.comp.encryption.general
Date: Friday 31st October 2008 18:10:00 UTC (over 8 years ago)
http://article.gmane.org/gmane.comp.encryption.general/12588/

Nakamoto, Satoshi (2008-10-31).
Bitcoin: A Peer-to-Peer Electronic Cash System
https://web.archive.org/web/20100704213649/https://bitcoin.org/bitcoin.pdf

Bakhtiari, S.; Safavi-Naini, R.; Pieprzyk, J.
Cryptographic Hash Functions: A Survey.
Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995.
ftp://ftp.cs.uow.edu.au/pub/papers/1995/tr-95-09.ps.Z.

Russell, A.
Necessary and Sufficient Conditions for Collision-Free Hashing
```

In Abstracts of Crypto 92. pp. 10-22-10-27, 1992.
ftp://theory.lcs.mit.edu/pub/people/acr/hash.ps.

Barak, B.; Arora, S.
Computational Complexity: A Modern Approach
Cambridge University Press
http://theory.cs.princeton.edu/complexity/