Some mathematical topics in blockchain and digital ledger technology

Christopher King
Department of Mathematics
Northeastern University
Boston MA 02115 USA.

August 21, 2022

1 Introduction

Digital ledger technology (DLT) spans a vast array of topics in computer science and business, including cryptocurrencies, digital security, smart contracts, decentralized finance and numerous other applications. There are many survey and review papers which can be consulted for a broad overview of DLT and related ideas. The goal of this paper is to focus on some mathematical themes which underlie DLT, and also to indicate some newer topics which have stimulated recent mathematical analysis. One of the mathematical themes we address is the topic of one-way functions, and we show how this idea is used to provide security protocols for Bitcoin. We also describe a mathematical algorithm which is being used on digital exchanges (DEXes) to facilitate automatic trading, thereby providing profit opportunities for cryptocurrency investors. Finally we discuss some new graph-theoretic ideas related to the use of directed acyclic graphs (DAGs) in place of blockchain.

The paper is organized as follows. We begin with a discussion of the notion of one-way functions, and then describe how this notion is implemented through the SHA256 hashing algorithm to provide security for the Bitcoin blockchain. We also review how one-way functions arise in the elliptic curve digital signature algorithm ECDSA. Then we describe how automatic market makers (AMMs) make use of a constant product algorithm to allow trades between digital assets. Finally we discuss some new approaches to DLT which employ DAGs in place of blockchain, and we review some mathematical results which have been obtained for the Tangle protocol used in the IOTA cryptocurrency.

2 Mathematical background: one-way functions

Loosely speaking, a one-way function is a map from bit strings to bit strings which can be computed efficiently, but whose inverse cannot be computed efficiently (see for example [9] for details and definitions). 'Efficiently' means that the time needed to compute the output grows polynomially with the length of the input string. Practically, this one-way property means that in order to produce

an output string with a given form it is necessary to perform a brute force search over the input space.

The mathematical description [11] refers to a family of functions, one for each length n of the input string. If n is the length of the input string and f_n is the function, then a pseudoinverse F_n would satisfy

$$f_n(F_n(f_n(x))) = f_n(x) \qquad \text{for all input } n\text{-bit strings } x. \tag{1}$$

Note that F_n need not be a one-to-one function; it just needs to provide some input string which can match a given output. Let $f = \{f_n\}_{n=1}^{\infty}$, $F = \{F_n\}_{n=1}^{\infty}$. The average case hardness definition of a one-way function f states that for any constant c and for any sequence of pseudoinverse functions F which can be computed in time which scales polynomially with n, for all n sufficiently large the relation (1) holds for at most a fraction n^{-c} of input strings. Note that the functions F may include randomized algorithms.

It is not known if a one-way function exists in this mathematical sense. In fact the existence of a one-way function would imply [11] that $P \neq NP$! The functions which are used in cryptographic protocols (some of which are described below) are believed to manifest this property, in the sense that it is believed that in order to find the pseudoinverse it is necessary to perform a brute force trial and error search over the input space. The functions used in practical applications are too complicated to allow a mathematical analysis which could prove this assertion, so their security ratings are based on long experience, and based on the fact that they have successfully resisted all attempts so far at efficiently finding an inverse. (In the future it may be possible for a quantum computer to shorten the trial and error search time by applying Grover's algorithm or something similar, but that seems a long way off at this point).

3 Bitcoin

Perhaps Bitcoin's mission can be best expressed by saying that it allows two strangers to remotely and directly send and receive cryptocurrency without needing a bank in the middle of the transaction. Bitcoin launched in 2009 and has since processed over 750 million transactions [6]. The core protocol of Bitcoin has never been hacked (there have been several hacks which exploited bugs in the implementation, but they were quickly corrected without lasting damage). The most remarkable feature of the Bitcoin blockchain is that its existence implies its authenticity; this feature will be elucidated in the following sections. However this stellar record has come at a price: it is estimated that Bitcoin mining operations consume over 100 TeraWatts, which is comparable to the total power consumed by Norway. Transactions are processed at the rate of 1000-2000 every 10 minutes, which corresponds to several per second, and this rate can never increase.

3.1 Bitcoin's one-way function: SHA256

If the success of an algorithm could be measured by the frequency with which it is run on a computer, then the SHA256 algorithm would surely be among the most spectacularly successful algorithms in history. The algorithm is run approximately 2×10^{20} times every second by the Bitcoin miners. Of course this is also a monument to excess and represents a colossal waste of energy and resources. However Bitcoin is here to stay, so it is interesting to understand how this algorithm operates.

The algorithm SHA256 was designed and published by the National Security Agency in 2001. It is an example of a hashing function. In general a hashing function maps longer bit strings to shorter bit strings, and so is a kind of compression algorithm. SHA256 has the following three useful properties:

- It maps an input string of arbitrary length to an output 256 bit string.
- It scrambles the input data. More precisely, if two input strings differ in any way, then the two output strings are completely different (of course this is an empirical statement based on observation and experience). So it mimics a random function in the sense that the output appears to be a random 256 bit string.
- It is conjectured that SHA256 behaves like a one-way function; again this is based on observation and experience. In practice this means that brute force searching is the only way to find the inverse.

The Bitcoin protocol employs SHA256 in several key ways, as part of the security protocol. Every block in Bitcoin has a unique identifier, presented as a 256 bit string. This ID is the output of the SHA256 algorithm. The input contains the ID of the previous block in the blockchain, and this linking between blocks is important for resilience again hackers. The security feature is contained in the startling form of the ID, which is a 256 bit number that starts with about 72 zeroes! Of course this is an extremely unlikely value to be found by chance, so its very existence implies a massive amount of work by the community of miners who use brute force random searching over the input space in order to force the SHA256 function to produce such an output. This repeated searching for suitable inputs for SHA256 is called the Proof of Work (POW) and is the largest factor contributing to the massive workload of Bitcoin mining. This interlinking property of the block IDs (which can be easily verified by any user) is the authenticity guarantee for the blockchain – it would be impossible to produce such a string of IDs without performing the massive random searches that the miners carry out (currently at the rate of 2×10^{20} times every second, as noted before).

The actual SHA256 algorithm works as follows. The input string is divided into blocks of 512 bits each (with padding by zeroes if needed). Then the algorithm takes each block and applies a variety of stretching, chopping, twisting and reversing steps which effectively randomize the string. The result is then combined with the second block where the same random mixing is applied, and so on up to the last 512 bit block. Everything is then repeated 63 more times. The final output 256 bit string is completely determined by the input (every step is deterministic!) but the mixing is so thorough that any change in the input will produce a completely different output. When this is applied to the block for Bitcoin the input consists of the previous block ID (256 bits), the Merkle tree hash of the transactions in the new block (256 bits), the difficulty (32 bits), the time stamp (32 bits) and the random nonce (32 bits). The SHA256 algorithm is then applied twice to this input to get the block ID. (Note for those who want to know more: the nonce allows the miner to search through 2^{32} possible inputs in order to find a suitable output, without making any other alterations to the block content. The 'randomizing' property of SHA256 means that each of these 2^{32} values for the nonce will produce completely different outputs, thereby allowing a pseudo-uniform search over the input space. However this random nonce field is quite small, so the protocol allows an additional random nonce to be added to the coinbase transaction, which creates the new Bitcoin payment for the miner. Changing this nonce trickles up to cause a change in the Merkle tree hash of transactions, and thereby leads to a new range of inputs for the random search).

Special purpose ASIC machines are used by Bitcoin miners to compute the SHA256 hashes for blocks. Each machine can perform up to 10^{14} hashes per second, so Bitcoin mining is no longer a place for amateurs!

4 Digital signatures

Bitcoin uses digital signatures as part of its security protocol. A Bitcoin transaction has one or more inputs and one or more outputs. Each input points to an output in a previous transaction on the blockchain, and imports the Bitcoins in that output. The previous output is owned by a Bitcoin address, and the instruction in the new transaction to transfer ownership of the Bitcoin must be digitally signed by the owner of that address, using their secret key (this is the private key which you never want to lose). The signor uses a digital signature algorithm which combines the data with their private key, to produce an output which can be publicly shared. The data and the signature are both made public, as well as the signor's public key. A witness may then run another algorithm on the data and signature which together with the signor's public key either verifies or fails to verify the signature. This algorithm is a one-way function, meaning that without knowing the private key it is very difficult to find a signature which will pass the verification step when combined with the data. So the security rests on the difficulty of forging the signature.

4.1 Elliptic curve digital signature algorithm: ECDSA

Bitcoin uses the elliptic curve digital signature algorithm (ECDSA) to produce the digital signature for spending transactions. An elliptic curve is the locus of points satisfying an equation of the form $y^2 = x^3 + ax + b$. The special properties of the elliptic curve allow an abelian group structure to be defined on the points of the curve (together with an identity point O at infinity). That is, given points P and Q on the curve, we can define a special addition operation so that $P+Q$ is another point on the curve, where the operation $+$ satisfies associativity. This is a geometric construction: the line containing P and Q intersects the elliptic curve at a unique point R, and we define $P + Q = -R$ where $-R$ is the reflection of R in the x-axis. The value $P + P$ is obtained in the limit where the connecting line becomes the tangent line at P. This allows integer multiplication to be defined, giving $n\,P$ for any integer n.

The geometric construction of $P+Q$ can be reformulated algebraically in a fairly simple way. This algebraic formulation extends to the case where the underlying field of real (or complex) numbers is replaced by a finite field. ECDSA uses the elliptic curve $y^2 = x^3 + 7$ over a finite field with modulus $r \simeq 2^{256}$ (meaning that calculations are done over the integers modulo r). Using the algebraic formulation it is straightforward to compute the integer multiple $Q = n\,P$ for any point P and integer n. However there is no known shortcut to invert this operation, that is to find n when given the points P and Q. The only known method is brute force search by trial and error. (The situation is similar to the discrete log problem, where one is given integers m, n and p, and one wants to find integer k so that $m^k = n$ modulo p). So this 'modular division' operation provides the one-way function for ECDSA, and thereby underpins the security of the ECDSA. Of course this is not proven to be a one-way function, but the method has a very strong track record of resisting attack.

5 AMM application on top of blockchain

Ethereum [3] is a very successful cryptocurrency due in large part to its use as a platform where applications can store and execute smart contracts. This facility has allowed digital exchanges (DEXes) to flourish. We will look in detail at how some automated market makers (AMMs) employ a constant product rule to service traders.

A constant product AMM (for example UNISWAP [1]) maintains two pools of assets, which we will call assets A and B. Often these assets are two cryptocurrencies. A trader can submit an offer to exchange some amount of asset A for some amount of asset B. For the moment we assume that no fees are charged in order to first explain the basic idea. Later we will introduce fees to show how the trading works in practice. The AMM sets the relative values of A and B by using the constant product rule. We define

$$\begin{aligned} X(t) &= \text{amount of asset } A \text{ in AMM pool at time } t \\ Y(t) &= \text{amount of asset } B \text{ in AMM pool at time } t \end{aligned}$$

The constant product rule is

$$X(t)\,Y(t) = k \tag{2}$$

where k is some constant that is determined by the AMM (depending on initial investments in the pools). The trader submits an offer to trade amount a of asset A for some amount of asset B. The AMM uses the constant product rule to compute the amount b of asset B corresponding to amount a of asset A:

$$(X(t) + a)\ (Y(t) - b) = k,$$

so the amount of asset B for this trade is

$$b = g_t(a) = \frac{aY(t)}{X(t) + a} = \frac{ak}{X(t)(X(t) + a)}.$$

So when the AMM executes the trade it returns the amount $g_t(a)$ of asset B to the trader. It then updates the amounts in its pools to the new values

$$\begin{aligned} X'(t) &= X(t) + a \\ Y'(t) &= Y(t) - g_t(a) \end{aligned}$$

It is reasonable to suppose that traders will take advantage of arbitrage opportunities to keep the the exchange rate offered by the AMM between assets A and B close to the market rate.

However this protocol is too rigid for practical trading. The trader's offer is sent as a transaction through the underlying smart contract platform which hosts the AMM (for example Ethereum), and so it enters a mempool where the miners select transactions for their blocks. Therefore there will certainly be a time delay before the offer arrives at the AMM, and thus the values of X and Y will have probably changed in the meantime. Therefore the trader will receive the amount $g_{t'}(a)$ of asset B (where t' is the time when the AMM processes the offer), and this could be much smaller than

their expected return $g_t(a)$. So the trader also provides a slippage value s in their offer. Then the AMM will accept the offer if

$$g_{t'}(a) \geq (1-s)\, g_t(a)$$

(the trader's offer contains a timestamp so the AMM knows the time t when it was created). This slippage allows enough flexibility so that the AMM can provide trading service.

Now we describe how trading fees operate. The AMM obtains its liquidity pools from investors who deposit amounts of assets A and B. The investors receive profits through fees which are charged for every transaction. (Note that these AMM fees are distinct from the gas fee which is charged by Ethereum). Let f be the fee charged by the AMM. When a trader offers an amount a of asset X, the AMM first subtracts a fraction fa before calculating the amount of asset B which will be returned to the trader. So the AMM calculates a new function $h_t(a)$ using the constant product rule for the amount $(1-f)\, a$:

$$h_t(a) = g_t\left((1-f)\, a\right) = \frac{(1-f)aY(t)}{X(t)+(1-f)a}$$

After this trade the amounts of assets in pools A and B are updated to the new values

$$\begin{aligned} X'(t) &= X(t)+a \\ Y'(t) &= Y(t)-h_t(a) \end{aligned}$$

As a consequence the constant on the right side of the product rule (2) also gets updated to a new value

$$k' = k\,\frac{X(t)+a}{X(t)+(1-f)a} > k.$$

This increase in k represents an increase in total liquidity, and reflects the profit which is being generated for the investors (the investors receive a pro-rated share of every transaction fee).

The AMM model is an active field of current research in the broader topic of decentralized finance (DeFi) (see for example [10] for an analysis of front-running attacks which exploit the slippage factor).

6 DAG extensions of blockchain

The architecture of the blockchain database is quite simple. As described above, the 256 bit ID of a block is created at the proof of work step by hashing the contents of the block together with the ID of its predecessor block. This inextricably links the ID's of the two neighboring blocks, and can be represented by a directed edge between the blocks. The result is a directed linear graph whose vertices are blocks with edges connecting nearest neighbors. Several cryptocurrencies (including Obyte, IOTA and Nano) have been created around the idea that additional complexity in the graphical structure of the database can provide additional security. Specifically, the idea is to create additional directed edges between vertices by including more than one 'parent' block ID when performing the POW step for a new block. The result is a directed acyclic graph (DAG).

6.1 The Tangle

Here we focus on the IOTA proposal, which is known as the Tangle [8]. In its original formulation, the Tangle is created directly by users who add individual transactions to create vertices on the DAG (unlike the blocks in blockchain which are created by miners and which each contain many transactions). The new vertex in the Tangle must be connected to two existing vertices on the Tangle. The protocol uses a modest POW for each connection (much smaller than in Bitcoin) which hashes together the ID's of these two parent vertices with the data in the transaction to create the new ID. This process of validation also checks consistency of the two parent vertices, thereby conferring trust to the parent nodes in the DAG. Further validations that indirectly link back then add to this trust. So security for a transaction accrues through the accumulated weight of POW from all transactions that either directly or indirectly link to it through their POW hashing. The idea is that an attacker would need unreasonably large resources in order to build a parallel DAG that could outcompete the accumulated weight of approvals. The original goals of the proposal included elimination of transaction fees and swift approval of transactions.

6.2 The random DAG process

The mathematical formulation of the Tangle leads to a very interesting random DAG-valued stochastic process where the graph grows through the attachment of new vertices correpsonding to transactions. Randomness enters the model through the arrival process for new transactions, the random parent vertex selection process, and the random delay time for the POW step.

The resulting graph process depends crucially on the method by which new transactions select two existing transactions for approval. One of the original proposals was the random tip selection algorithm, meaning that two leaves of the DAG (known as tips in the Tangle literature) are randomly and uniformly selected by each newly arrived transaction, and POW is carried out on them. This method has some security drawbacks [8], but it also has some advantages regarding the elimination of so-called 'orphan transactions' [4]. In the paper [8] it was shown that in steady state the number of tips L in the Tangle (again tips are the same as leaves, that is vertices with zero in-degree) has mean value

$$L = 2\,\lambda\,h$$

where λ is the arrival rate of new transactions, and h is the mean delay time for the POW step (recall that h is the time between the selection of parent vertices by a new arrival and its eventual addition as a new vertex to the Tangle). The Tangle model with random tip selection and constant delay time h was analyzed in [5] where it was shown that the graph process is ergodic and converges to a unique stationary distribution, and that the fluid limit (where arrival rate λ diverges) is described by a stochastoc delay differential equation. The random tip selection process with bivariate delay time distribution was analyzed in [7], and a formula obtained for the mean number of tips in steady state. In another direction the recent paper [2] explores the asynchronous composition model where the delay times are IID with general distributions. The authors prove ergodicity of the models, and also prove results about one-endedness which are highly relevant for consensus of the ledger.

References

[1] Angeris, G., Kao. H.-T., Chiang, R., Noyes, C., Chitra, T., "An analysis of Uniswap markets", Cryptoeconomic Systems Journal, 2019.

[2] Dey, P.S. and Gopalan, A., "On an Asymptotic Criterion for Blockchain Design: The Asynchronous Composition Model", arXiv:2202.05080 [math.PR] (2022).

[3] Wood, G., "Ethereum: A Secure Decentralised Generalised Transaction Ledger"; Buterin, V., "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform", available at https://ethereum.org/en/whitepaper/.

[4] Ferraro, P., King, C., and Shorten. R, "On the stability of unverified transactions in a DAG-based Distributed Ledger", IEEE Transactions on Automatic Control, Vol. 65, Issue 9, 2020.

[5] King, C., "The fluid limit of a random graph model for a shared ledger", Advances in Applied Probability, Volume 53, pp. 81 - 106 (2021).

[6] Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system", available at https://bitcoin.org/bitcoin.pdf, 2008.

[7] Penzkofer, A., Saa, O. and Dziubatowska, D., "Impact of delay classes on the data structure in IOTA", arXiv:2110.06003 [cs.DC] (2021).

[8] Popov, S., "The Tangle-Version 1.4.2", available at https://iota.org/IOTA_Whitepaper.pdf.

[9] Sipser, M., "Introduction to the Theory of Computation", PWS Publishing, Boston, 1997.

[10] Heimbach, L., Wattenhofer, R., "Eliminating Sandwich Attacks with the Help of Game Theory", arXiv:2202.03762 [cs.GT], 2022.

[11] Wikipedia, https://en.wikipedia.org/wiki/One-way_function.