Before the First Block: A Historical Overview of Distributed Consensus and Cryptography

Stephen Corya
Corya Enterprises, LLC
Indianapolis, Indiana
stephen@corya.co

"Who knows what will be the future incarnations of money? Computer bytes?"
—Milton Friedman, *Money Mischief: Episodes in Monetary History*, 1992

***Abstract*—Describing a blockchain as "a chain of blocks" does little to enhance the layman's understanding. While the first bitcoins were created in 2009, there is a great deal of preceding work which contributed to this development. This article presents a brief history of distributed consensus and cryptography with the hope that the reader may better understand blockchain technology and its significance.**

***Index Terms*—Blockchain, cryptographic protocols, distributed databases, proof of stake, proof of work**

I. Distributed Systems

A blockchain is a type of distributed computer system. Advantages of distributed systems (compared to single, separate computers) include reliability, flexibility, and geographical distribution [1]—all essential components of a highly available, global system such as a blockchain. For the sake of this article, a distributed system will be defined as a collection of computer processes connected by a communications network. A shared state among the processes distinguishes such a collection as a system, beyond its constituent components. Data are among the components of this shared state.

A. Consensus and Replication

Distributed consensus is the agreement upon a proposed datum to be added to shared state by a distributed system. Properties of consensus in the context of computer systems include safety (every process agrees on the proposed datum), liveliness (the processes come to an agreement), and fault-tolerance (the system as a whole can continue its operation in spite of potential, individual process failure) [2].

Shared data must be replicated to provide high availability. Storing data on a single computer creates a single point of failure; if that computer goes offline, the entire networked system effectively collapses. Automatically reaching consensus can be achieved through algorithms, which may be formalized as protocols.

B. Synchrony

Distributed systems can broadly be categorized as either synchronous or asynchronous. In a synchronous system, the constituent processes agree on some timing constraint. In practice, over a wide-area network such as the Internet, two networked processes operating on different hardware in separate geographic regions can only agree upon timing to a degree. Dwork, Lynch, and Stockmeyer define synchrony within such a degree as "partially synchronous" in 1988 [3]. Many distributed consensus algorithms rely on partial synchrony.

II. Paxos

Paxos is among the most thoroughly researched consensus algorithms [4]. While some regard the Paxos algorithm as difficult to understand [5], its original presentation as a metaphor in 1998 [6] may be helpful to the non-geek. The Paxos algorithm is named after the Greek Island of Paxos and explained through a fictional parliamentary system therein.

A. Constraints

The parliament requires a consistency of its record (passed legislation) in spite of its members' "frequent forays from the chamber and the forgetfulness of their messengers." Messengers are liable to submit their messages more than once or not send their messages at all, and legislators can only communicate by messenger.

In addition to strict consistency of their legislative record, the Paxons require that a decree be passed if there is a quorum of legislators within the Chamber for a "sufficiently long period of time"; they may not simply agree to leave the record blank. The requisite period of time in the Chamber is measured by each legislator's hourglass.

B. The Protocol

To add a decree to the record, each legislator maintains a ledger, wherein he records a numbered sequence of passed decrees. Decrees cannot be changed once recorded. From an example in [6], a legislator has the entry "155: The olive tax is 3 drachmas per ton" on her ledger. No other legislator may have a different decree in his ledger at sequence number 155; although, another legislator could have no entry at sequence 155 if he has not yet received his message. Legislators additionally maintain notes on the back of their ledgers. These notes can be crossed out.

Each proposed decree is accompanied by a ballot issued by an elected president. With a quorum present in the Chamber for a certain time, exactly one legislator will be elected president. Any member of parliament within the Chamber may propose a ballot to the president. All legislators within the Chamber vote in the affirmative to all proposals. These ballots are numbered, and the president chooses a number greater than the last recorded decree.

C. Applicability

In the analogy, departures from the parliamentary Chamber represent the failure of processes within a distributed system. The messengers represent a communications network among the processes. The hourglasses represent a requirement for (at least partial) synchrony. One may note that these messengers "could be counted on not to garble messages", and all the legislators are always adherent to the parliamentary protocols so long as they are in the Chamber.

III. The Byzantine Generals Problem

Fischer [7] characterizes two types of process failures: crashes and Byzantine failure. If a process crashes, it ceases to communicate with other processes in the system. In the case of a Byzantine failure, which takes its name from the Byzantine Generals Problem [8], a process sends deviant information to the other processes. The problem can be represented in terms of "loyal generals" and "traitors". The loyalists must agree on a plan, and this plan must have sufficient support among a number of loyalists to succeed, regardless of the traitors. Characterizing Byzantine failure with Paxos terminology: a messenger is sending "[garbled] messages", a legislator is acting in violation the protocol, or both.

Dolev expands upon the Byzantine Generals Problem in [9]. He concludes that consensus is possible, even in the case of Byzantine failure, so long as less than one third of the processes in the system are faulty at and least half of the processes may reach one another.

A. (Practical) Byzantine Fault Tolerance

Castro and Liskov present a solution to the Byzantine Generals Problem in 1999 [10]. Their solution (Practical Byzantine Fault Tolerance or "PBFT") is notable for several reasons. Among these: it does not require network synchrony, and it ensures both safety and liveliness, so long as there is a sufficient number of reliable processes (one third, as shown by Dolev) within the system. The practicality of PBFT is in part a consequence of its significant reduction in the number of messages that must be exchanged by distributed processes in comparison to earlier solutions.

IV. Mutual Suspicion and Security

The Byzantine Generals Problem necessitates suspicion among the distributed processes. In [11], Chaum details algorithms which can be used to establish trust in a computer system by parties that may or may not trust one another. He provides an example of a computer system that tracks a bank's checking account balances. A group maintaining this system or a similar system is referred to as a group of "trustees." Data are stored in "vaults" such that other parties can access and verify them.

Chaum's solution leverages public key cryptography to establish trust among the trustees and encrypt communications.

A. Public Key Cryptography

Public key cryptography as employed by Chaum's algorithms is introduced, as a theory, by J. H. Ellis in 1970 [12]. In 1973, Clifford Cocks [13] provides a possible solution to Ellis' theory. These men worked at the Government Communications Headquarters of the United Kingdom, and their work was classified until 1997 [14]. Essential to public key cryptography is public key exchange, whereby two parties agree upon which encryption ciphers they will use without previously communicating in confidence.

Due to the secrecy of Ellis' and Cocks' work, credit for the invention of public key exchange is often given to Diffie and Hellman. In 1976, they publish [15]; however, in 1974, Ralph Merkle (a University of California, Berkeley, undergraduate student at the time) presents [16] to Professor Lance Hoffman, a project proposal for his CS244 class. Like Ellis, Merkle contends that public key exchange is possible, and suggests that it would make a suitable undergraduate project. Hoffman rejects the proposal, and Merkle drops the class [17]. In [18], Hellman suggests that what is commonly called "Diffie-Hellman" exchange be called “Diffie-Hellman-Merkle Key Exchange."

Paramount to all cryptography is the knowledge of keys. Prior to Diffie-Hellman, these keys needed to be shared securely via a separate communication channel. That is, in order for two or more parties to privately communicate over a public network such as the Internet, they had to exchange keys ahead of time over a private network.

Merkle provides an eloquent explanation in [19]. He describes three parties X, Y, and Z. X and Y wish to communicate securely without Z being able to decipher their messages.

"X and Y must both know what the key is, and must insure that Z does not know what it is. In the traditional paradigm for cryptography, this situation comes about by the transmission of the key from X to Y over some special and secure communications channel, which we shall refer to as the key channel. Z cannot intercept messages sent on this channel, and the key is therefore safe.

"The reason that the key channel is not used for normal communications is because of its expense and inconvenience. Radio and telephone cannot be used, as both are vulnerable to passive eavesdropping. Registered mail might be acceptable for moderate security. Word of mouth is better, but listening devices might compromise it. Perhaps the only safe method is to send a trusted courier, with an attaché case chained to his wrist. This requires that you trust the courier. Whatever the method used, if Z should manage to discover the key by 'practical cryptanalysis,' then X and Y might very well continue in blissful ignorance of the fact."

A practical analogy of public key cryptography can be expressed in terms of a P. O. box. A public key is analogous to the P. O. box's address. Anyone can send a letter to the box; however, a private key (the key that unlocks the box) is required to read the letters. The analogy can be stretched to explain another important aspect of public key cryptography: signed messages. Assuming that any party that can send a letter has a unique signature, a letter's origin can be verified upon its reception based on its signature. A letter cannot be practically unsigned; signing a letter is a one-way function. The analogy is stretched, because in public key cryptography, signing is done with private keys. The creation and application of a unique, wax seal with the imprint of the P. O. box’s key may create a more accurate illustration.

V. Hashing

Hash algorithms (or hash functions) are another type of one-way functions, with applications in both distributed consensus and cryptography. Applying a hash algorithm is commonly referred to as "hashing". One of the first hash algorithms MD2 (Message-Digest) is put forth by Kaliski in 1992 [19]. Hash algorithms produce a fixed-length digest (a "hash") of a given input. For example, the MD2 hash of the text "for example" is

5b14d4e48ab3f0a803daff2ff53d36ba.

It is not feasible to derive input text from its hash. Other notable hash algorithms include SHA-1 (Secure Hash Algorithm 1), published by the National Institute of Standards and Technology in 1993 [20], and SHA-2, published in 2002 [21]. Correia et al. [22] puts forward an approach to Byzantine-tolerant distributed

consensus that utilizes hash algorithms as part of its methodology.

VI. Blockchains

Correia [23] describes a blockchain as "an exciting new technology" which "is essentially a secure, unmodifiable, append-only, log of transactions." This log of transactions can be summarized as a ledger. Such a ledger is often used to maintain a record of monetary units possessed by a number of actors. These monetary units are often called "cryptocurrency" or "coins." One may note that while cryptocurrency and blockchains are related, the nature of the relationship is such that blockchain technology enables cryptocurrency. Constituent, distributed processes are often termed "nodes" in the context of blockchain systems.

A. Bedrock

Bedrock [24] is a distributed database consensus software, started in 2007; its initial deployment may be considered the start of the first blockchain. Like many other consensus protocols, Bedrock is designed to work over a wide-area network such as the Internet. Unlike many other blockchain protocols, Bedrock is not designed to reach consensus among hundreds or thousands of nodes. Its authors describe Bedrock as a "'private' blockchain" operating among a "small cluster of servers" [25]. These properties make Bedrock a fine ,explanatory blockchain.

"Under the hood", Bedrock maintains transactions within a table, separately on every node. The transactions have three properties. The first property is an ID—a simple, unique numeral assignment for each transaction. The second property is a query, which can be considered a modification instruction for the system's data. The table's third property is a SHA-1 hash of both the current query and the previous query's hash. By hashing both the current query and the previous query’s hash together into a single, new hash that gets added to the table, each transaction's hash property is a product of both itself and its predecessor; the transactions are "chained" together.

When a node connects to the system, it broadcasts the ID and SHA-1 hash of the latest transaction in its table. If another node has this same transaction in their own table, these nodes are in consensus up to this transaction ("block") in the chain. Because transaction ID's are incremental, one node's latest transaction ID may be lower than another's. In this case, transactions can be synchronized and their queries committed to the local state of each node.

In some cases, different nodes may disagree on which transaction corresponds a given ID. These incongruous nodes may be described as "forked." In this state, the nodes will refuse each other's attempted communications. Multiple nodes can exist on each fork. A Paxos-based election protocol will ensure that only one of these forks will elect a new leader, based on which fork has a quorum. As with proposals in Paxos, all transactions are sent to the leader.

B. Proof of Work

All algorithms and protocols reviewed heretofore have no cost associated with proposing or committing transactions. In Paxos terms, there is no fee associated with adding to a ledger or passing legislation. This is the case with innumerable, additional protocols. In 1992, Dwork and Naor [26] put forth a possible technique to create a computational cost to modifying the state of a computer system. Their work is inspired by a glut of junk email, and this remains a theme throughout the paper; however, their technique can be extended as an "access control mechanism that can be used whenever it is desirable to restrain, but not prohibit, access to a resource."

Dwork and Naor propose a "pricing function" to control the cost of various digital activities, such as sending emails. In their model, the recipient of the email could easily discern whether or not the pricing function had been executed. Multiple possible functions are explored, and the authors enumerate three categories of these functions: easy, moderate, and hard. They conclude that a moderately difficult pricing function would be ideal, but at the time, there was "no theory of moderately hard functions."

C. Cryptocurrency

Transitioning from email to money, Wei Dai publishes b-money in 1998 [27]. He proposes that money can be created in a distributed system "by broadcasting the solution to a previously unsolved computational problem." His approach rewards the money creator based on the difficulty of the computational problem, depending on hypothetical market conditions.

Satoshi Nakamoto (possibly a pseudonym [28]) creates the first practical cryptocurrency in 2008 with the publication of [29], and he calls this new currency “bitcoin”. This publication is widely regarded as the creation of the term "blockchain", wherein Nakamoto states plainly, "blocks are chained". The first bitcoin block, known as the "genesis block", is created in 2009.

Cryptocurrencies are fundamentally dependent on consensus; every node must agree on which actors control how many units of currency. In his novel blockchain protocol, Nakamoto addresses the issue of “moderately hard functions” by varying the difficulty of finding a correct hash for a block of transactions. The variability is dynamic. As more computational resources are allocated to the creation of new blocks—and new bitcoins, the difficulty increases. Where Bedrock uses an election to determine which forked nodes have a quorum, bitcoin takes another approach: the longest chain is assumed to be correct.

Since bitcoin is designed to be implemented on mutually suspicious nodes, additional differentiation from Bedrock is required. When a user submits a bitcoin transaction, a hash is calculated from both the hash of the previous transaction and the public key of the recipient. The user submitting the transaction signs this hash with a private key. Using this signature, the recipient is able to verify the transaction; however, this does not in and of itself stop the submitter from sending the same coins to multiple recipients. This problem is referred to as "double spending."

The double spending problem is addressed by putting a timestamp on each block, where a block is a collection of transactions. Bitcoin uses a distributed timestamp server. The timestamp server calculates the hash of a block, and then broadcasts that hash to other nodes. A block's hash also includes its predecessor's hash, thus the blocks are chained together. Because the timestamp server nodes are mutually suspicious, bitcoin employs a proof of work mechanism. Each block of transactions contains an incremental number called a "nonce". A network of actors competes to complete the proof of work required to find a hash for a new block that satisfies the conditions set by the dynamic difficulty. This competition is often referred to as "mining". Miners are rewarded for their computational effort with special transactions, the creation of new bitcoins.

An overview of the bitcoin network’s functionality, directly from [29]:

1. New transactions are broadcast to all nodes.

2. Each node collects new transactions into a block.

3. Each node works on finding a difficult proof of work for its block.

4. When a node finds a proof-of-work, it broadcasts the block to all nodes.

5. Nodes accept the block only if all transactions in it are valid and not already spent.

6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

One may note the assumption that every node in the network will participate in the proof of work competition. In light of the extraordinary amount of computing power dedicated to mining [30], many nodes elect not to participate at all. These non-mining nodes still keep a record of the blockchain, but they do not propose new blocks. Both mining and non-mining nodes are free to join or leave the network at any time; they synchronize with the longest chain accordingly.

D. Proof of Stake

Proof of stake is a consensus protocol that builds upon the foundation laid by proof of work, likewise applicable to cryptocurrency. This novel protocol is formally introduced in 2012 by Sunny King and Scott Nadal [31]. (At some point, the name of their original proof of stake cryptocurrency "PPCoin" was superseded by "Peercoin".) The authors recognize the inherent resource consumption required by an effective proof of work system, and subsequently put forth a concept they call "coin age". To illustrate through example, "if Bob received 10 coins from Alice and held it for 90 days, we say that Bob has accumulated 900 coin-days of coin age." To accurately calculate coin age, naturally each individual transaction requires a timestamp.

As with bitcoin, the acceptance of a new block by other nodes rewards its proposer with newly created coins. In order to earn the privilege of proposing a new block, a Peercoin holder sends coins to himself; in the process, he consumes some portion

of his own coin age. The new block must meet a hash requirement, as is the case is proof of work; however, this requirement is significantly easier to compute than the same in a proof of work system. This reduced hash difficulty contributes to proof of stake's diminished resource requirements. Even though the hash difficulty is relatively low, its computation can be regarded as moderately difficult, and it fluctuates to accommodate more or less mining effort, as with proof of work. Resolution of fork conditions differs from bitcoin's proof of work. Instead of assuming the longer chain is correct, Peercoin assumes that the correct fork is the one with the greater consumed coin age.

Proof of work and proof of stake are the two most prominent consensus protocols for blockchain systems, powering bitcoin and Ethereum respectively. (The Ethereum blockchain was switched from proof of work to proof of stake on September 15, 2022 with an update commonly called "The Merge"; the update reduces energy consumption by ~99.95% [32].) These protocols represent milestones in the long, intertwined roads of distributed consensus and cryptography.

VII. Fin.

A blockchain is a chain of blocks.

References

[1] “Advantages and disadvantages of Distributed Systems,” GeeksforGeeks, https://www.geeksforgeeks.org/advantages-and-disadvantages-of-distributed-systems/ (accessed Mar. 30, 2025).

[2] A. Dinh, “History of the Impossibles - CAP and FLP,” https://dinhtta.github.io/flpcap/ (accessed Apr. 17, 2025).

[3] C. Dwork, N. Lynch, and L. Stockmeyer, “Consensus in the presence of partial synchrony,” *Journal of the ACM*, vol. 35, no. 2, pp. 288–323, Apr. 1988. doi:10.1145/42282.42283

[4] C. Cachin, "Yet another visit to paxos," IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 6, pp. 624-637, Nov.-Dec. 2015. Available: https://www.semanticscholar.org/paper/Yet-Another-Visit-to-Paxos-Cachin/84edec63e08ba4e5a6ad9029d86f93b420e79392. (accessed Apr. 17, 2025)

[5] Lamport, L. "Paxos made simple." Microsoft Research, 2001. https://www.microsoft.com/en-us/research/publication/paxos-made-simple/. (accessed Apr. 12, 2025).

[6] Lamport, L., "The part-time parliament," ACM Transactions on Computer Systems (TOCS), vol. 16, no. 2, pp. 133-169, May 1998.

[7] M. J. Fischer, “The consensus problem in Unreliable Distributed Systems (a brief survey),” Lecture Notes in Computer Science, pp. 127–140, 1983. doi:10.1007/3-540-12689-9_99

[8] L. Lamport, R. Shostak, and M. Pease, “The Byzantine generals problem,” ACM Transactions on Programming Languages and Systems, vol. 4, no. 3, pp. 382–401, Jul. 1982. doi:10.1145/357172.357176

[9] D. Dolev, “The Byzantine Generals Strike again,” Journal of Algorithms, vol. 3, no. 1, pp. 14–30, Mar. 1982. doi:10.1016/0196-6774(82)90004-9

[10] Castro, M., & Liskov, B. (1999). Practical Byzantine Fault Tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation (pp. 173-186). New Orleans, USA.

[11] Chaum, D. L. (1982). Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups. University of California, Berkeley.

[12] J. H. Ellis, “The Possibility of Secure Non-secret Digital Encryption,” Government Communications Headquarters, Jan. 1970. Accessed: Apr. 08, 2025. [Online]. Available: https://cryptocellar.org/cesg/possnse.pdf

[13] C. C. Cocks, “A Note on ‘Non-secret Encryption,’” Government Communications Headquarters, Nov. 1973. Accessed: Apr. 08, 2025. [Online]. Available: https://cryptocellar.org/cesg/notense.pdf

[14] N. Smart, “Dr Clifford Cocks CB,” Honorary Graduates, https://www.bristol.ac.uk/alumni/our-alumni/honorary-degrees/honorary-graduates/2008/cocks.html (accessed Apr. 12, 2025).

[15] W. Diffie and M. Hellman, “New Directions in Cryptography,” IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644–654, Nov. 1976. doi:10.1109/tit.1976.1055638

[16] R. Merkle, “Project Proposal.” Ralph C. Merkle https://www.ralphmerkle.com/1974/FirstCS244projectProposal.pdf

[17] History of public key cryptography, https://www.ralphmerkle.com/1974/ (accessed Apr. 13, 2025).

[18] M. E. Hellman, “Cybersecurity, nuclear security, Alan Turing, and illogical logic,” ACM Turing award lectures, p. 2015, Oct. 2016. doi:10.1145/1283920.2976757

[19] B. S. Kaliski, “The MD2 Message-Digest Algorithm.” RSA Laboratories, Redwood City, CA, Apr. 1992 https://www.rfc-editor.org/rfc/pdfrfc/rfc1319.txt.pdf

[20] “FIPS Publication 180.” National Institute of Standards and Technology, Gaithersburg, MD, May. 11, 1993

[21] “FIPS Publication 180-2.” National Institute of Standards and Technology, Gaithersburg, MD, Aug. 1, 2002

[22] M. Correia, N. F. Neves, L. Lung, and P. Veríssimo, "Byzantine-Resistance Consensus based on a Novel Approach to Intrusion Tolerance," Repositório da Universidade de Lisboa, 2004. Available: http://hdl.handle.net/10451/15076 (accessed Apr. 14, 2025).

[23] M. Correia, “From Byzantine Consensus to Blockchain Consensus,” in Essentials of Blockchain Technology, Boca Raton, Florida: CRC Press, 2019

[24] “Bedrock – rock-solid distributed data,” Bedrock by Expensify, https://bedrockdb.com/ (accessed Apr. 14, 2025).

[25] “Blockchain – Bedrock’s secret sauce (before it was cool),” Bedrock by Expensify, https://bedrockdb.com/blockchain.html (accessed Apr. 14, 2025).

[26] C. Dwork and M. Naor, “Pricing via processing or combatting junk mail,” Lecture Notes in Computer Science, pp. 139–147, 1992. doi:10.1007/3-540-48071-4_10

[27] W. Dei, “b-money.” 1998 http://www.weidai.com/bmoney.txt

[28] S. Haig, “John McAfee is 99% certain he knows who satoshi nakamoto is,” Cointelegraph, https://cointelegraph.com/news/john-mcafee-knows-who-satoshi-nakamoto-is (accessed Apr. 15, 2025).

[29] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System.” Bitcoin Project, Oct. 2008 https://bitcoin.org/bitcoin.pdf

[30] M. Morey, G. McGrath, and H. Minato, “Tracking electricity consumption from U.S. cryptocurrency mining operations,” Today in Energy, https://www.eia.gov/todayinenergy/detail.php?id=61364 (accessed Apr. 18, 2025).

[31] S. King and S. Nadal, “PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake.” Peercoin Foundation, Amsterdam, Noord Holland, 2012 https://www.peercoin.net/papers/peercoin-paper.pdf

[32] “The merge,” Ethereum.org, https://ethereum.org/en/roadmap/merge/ (accessed Apr. 18, 2025).

