Bitcoin Protocol Mechanics for Economists: A Compact Reference

Diego R. Llanos
University of Valladolid, Spain

Javier Perote
University of Salamanca, Spain

José D. Vicente-Lorente
University of Salamanca, Spain

January 27, 2026

Abstract

Economists increasingly engage with Bitcoin's macroeconomic and financial implications, yet many debates implicitly assume protocol properties that are rarely stated with mechanical precision. This paper offers a concise technical primer on Bitcoin protocol mechanics, aimed at providing the minimal foundations for interpreting claims about decentralization, immutability, and a credible issuance cap without trusted intermediaries. We explain how transactions encode ownership and transfer via public-key cryptography and digital signatures, why the blockchain functions as an append-only, replicated ledger, and how full nodes, wallets, and miners jointly enforce validity and compliance with the rules. We then describe Proof-of-Work as a coordination and security mechanism, clarifying how confirmations deliver probabilistic finality and why rewriting settled history is computationally prohibitive. The paper also outlines Bitcoin's deterministic issuance schedule via the coinbase reward and halving rule, and briefly situates fee revenue and second-layer protocols as the long-run basis for payments and security as block subsidies decline. Three appendices provide a didactic treatment of transactions, a compact summary of cryptography, and an overview of the mining workflow.

Keywords: Bitcoin; blockchain; Proof-of-Work; transaction validation; monetary issuance; digital signatures.

1 Introduction

Bitcoin is routinely discussed in macroeconomics, finance, and policy as a prospective medium of exchange, store of value, or alternative monetary standard. Yet many of the strongest claims made on either side of these debates rest on protocol-level facts that are often treated as black boxes: how ownership is defined without named accounts, how validity is enforced without a central operator, why the transaction history is difficult to rewrite, and what it means for issuance to be credibly predetermined. For economists, a minimal but mechanically correct understanding of these building blocks is useful for two reasons. First, it helps distinguish institutional narratives from properties that are actually enforced by rules and verification. Second, it clarifies where economic trade-offs arise endogenously from the design, such as custody versus convenience, finality versus speed, and security versus resource expenditure.

This paper is a technical primer for readers who want protocol intuition precise enough to support economic interpretation, yet compact enough to serve as a reference. We focus on the canonical Bitcoin design: the transaction model that defines ownership and transfer; the blockchain data structure that records history; the network roles that produce and verify that history; and Proof-of-Work as the mechanism that both selects a single chain and raises the cost of rewriting it. We also explain why Bitcoin's issuance path is not a discretionary policy choice but rather a rule audited and enforced by validating nodes. The goal is descriptive rather than empirical: we do not estimate causal effects, assess welfare, or provide investment advice.

The remainder of the paper is organized as follows. Section 2 introduces transactions, signatures, and the notion of ownership in a system without account registries. Section 3 presents the blockchain as a replicated, append-only ledger and explains how hash-linking and Merkle commitments support integrity. Section 4 clarifies the distinct roles of full nodes, wallets, and miners in validating rules, managing keys, and proposing blocks. Section 5 describes how mining clears transactions and why block production is paced. Section 6 explains the sources of Bitcoin's immutability and the meaning of confirmations as probabilistic finality. Section 7 discusses how a new node can verify a downloaded chain from first principles rather than trusting a third party. Section 8 unpacks Proof-of-Work, including difficulty adjustment and security intuition, and contrasts it with Proof-of-Stake at a high level. Section 9 explains the issuance schedule, the halving rule, and the long-run shift toward fee-funded security and scaling layers.

To accommodate different entry points, the paper is complemented by three appendices. Appendix A provides a didactic account of how Bitcoin transactions and addresses work, including how signing authorizes spending. Appendix B summarizes the cryptographic primitives used throughout the protocol, such as hash functions, asymmetric encryption, and digital signatures. Appendix C sketches the mining workflow at an algorithmic level, highlighting the steps needed to assemble and propose a valid block. Readers already familiar with these elements can skip directly to the main sections and use the appendices as a compact lookup.

2 Transactions, Signatures and Ownership

To understand how Bitcoin enforces scarcity and property rights without a central authority, it is essential to examine its transaction model, address format, and signature mechanism. At its core, a Bitcoin transaction consists of one or more inputs (previously received coins) and one or more outputs (new recipients). Unlike conventional banking, all coins from an input are consumed entirely in a transaction—any leftover balance is returned to the sender as "change" via a new output.

Each Bitcoin address corresponds to a public key; the right to spend funds associated with that address depends on the corresponding private key. This asymmetric cryptography ensures that only the legitimate holder of a private key can authorize a transaction. Before a transaction is submitted to the network, each input is signed with the appropriate private key. Nodes verify these signatures and check that inputs are unspent, thus validating the transaction. Appendix A describes this process and its related technologies in detail.

Address generation begins with a secure random seed, which is used to produce private/public key pairs via elliptic curve cryptography (secp256k1). To enhance usability, the seed is often encoded as a 12- or 24-word phrase using the BIP39 standard. This phrase grants complete control over the associated Bitcoin funds; whoever possesses it can spend the coins. This cryptographic closure eliminates the need for an external registrar: possession of the private key *is* legal title. Appendix B offers a succinct yet precise explanation of the role of cryptography in Bitcoin.

This digital notion of ownership, where control is enforced by cryptography rather than legal designation, represents a fundamental innovation. Unlike conventional accounts held at institutions, Bitcoin addresses are pseudonymous and noncustodial. If the private key (or seed) is lost, the funds are irretrievable. If stolen, the rightful owner has no recourse. The Bitcoin protocol thus redefines possession and trust, grounding them in information theory rather than institutional frameworks.

Together, these mechanisms (inputs and outputs, public-key cryptography, digital signatures, and seed-based key generation) form the backbone of Bitcoin's decentralized system. They enable secure, censorship-resistant transfer of value without reliance on intermediaries.

3 The Bitcoin Blockchain

The way a Bitcoin transaction is built, as described in detail in Appendix A, eliminates the possibility that a malicious actor can change it undetected. In this section, we will introduce the Bitcoin blockchain, the data structure that stores all Bitcoin transactions.

In functional terms, the Bitcoin blockchain (or the "blockchain" for short) is a data structure that stores all transactions carried out since Bitcoin's inception. We can see it simply as a ledger. The blockchain possesses several unique characteristics. The first one is that thousands of copies of the

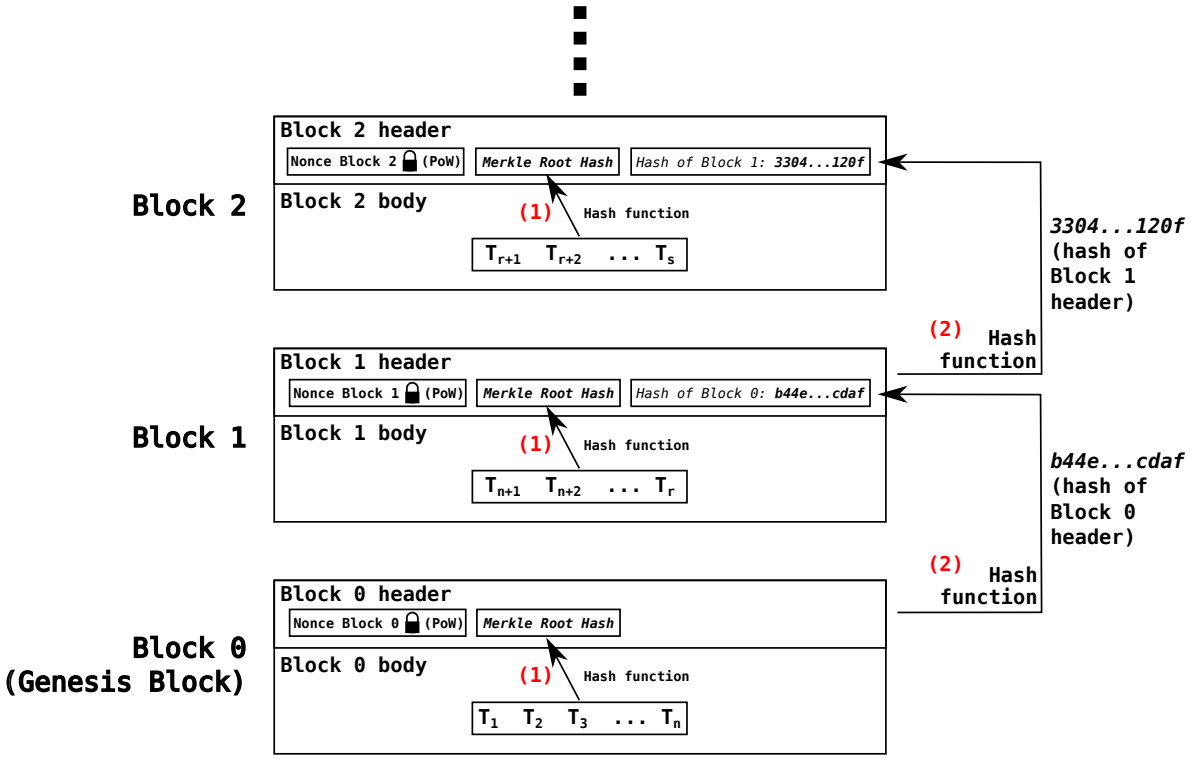

Figure 1: In the blockchain, the block header of each block above the Genesis Block includes the hash of the previous block header. If a block is changed, its new hash will not match the value stored in the block above it. Related concepts, such as the use of hash functions, are described in detail in Appendix B.

Bitcoin blockchain are scattered around the globe. This is why the blockchain is frequently referred to as a "distributed ledger". However, this name is somewhat misleading: It would be better to say "replicated ledger", because each copy is complete, that is, the blockchain is not "fragmented" in any way.

A second characteristic that makes the blockchain special is that it is an append-only data structure. On average, every ten minutes, a new chunk (or *block*) of transactions is appended to the existing blockchain. The general structure of the Bitcoin blockchain is depicted in Figure 1. Each new block of transactions is appended to the previous one. Cryptographic methods ensure that no block of transactions can be modified without altering all blocks built upon it, a task that would require an unfathomable amount of computing power.

At the time of writing, the Bitcoin blockchain comprises approximately 866,000 blocks, and the entire data structure occupies around 608 GB of disk space. It is perfectly possible to download and verify the Bitcoin blockchain using a PC or laptop. It may take from a couple of hours to several days, depending on the performance of our computer and the speed of our Internet connection. Fortunately, as we will see, there is no need to download a local copy of the blockchain to interact with it.

4 The Different Roles in the Bitcoin Network

The Bitcoin network comprises thousands of independent computers, each serving one of three primary roles: full nodes, wallets, or miners. These components operate in a decentralized manner, without any central authority or single point of control. There is no "Central Bitcoin Computer" whose deactivation could halt the system. In this sense, attempting to stop Bitcoin is analogous to trying to prevent people from playing chess: The rules are public, and anyone can replicate the game using their own board and pieces. Consequently, disrupting Bitcoin would require measures as extreme as disabling the Internet globally.

4.1 Full bitcoin nodes

There exist tens of thousands of *full Bitcoin nodes*, or simply "nodes", around the world. Each one of them stores a copy of the entire Bitcoin blockchain. When a new node is set up and asked to download the Bitcoin blockchain, it starts from block zero and requests other nodes for subsequent blocks in order, verifying that all information is correctly signed and that the blockchain structure is valid. Once it has received all the transaction blocks, the node is said to be "synchronized" with the network. (We will return to this point in Section 7.) Once the node is synchronized, it starts performing two different tasks at the same time:

- Each time a new pending transaction arrives at a Bitcoin node, the node verifies that the transaction is valid. This includes verifying that the inputs are valid (i.e., that the input addresses are associated with outputs from previous transactions) and that the transaction adheres to Bitcoin's rules (such as ensuring there is no double-spending and that the correct cryptographic signatures are present). If the transaction is valid, it is added to the node's *mempool*, a waiting area for pending (also known as *unconfirmed*) transactions. Once verified, the node also broadcasts this new transaction to other nodes in the network, which will verify it and add it to their mempools.

- Each time a new block of transactions arrives at a Bitcoin node to be added to the top of the blockchain, a situation that occurs every 10 minutes on average, the node checks that the new block is correct, stores the block on top of the existing blockchain, removes all the transactions included in that block from its unconfirmed transactions' mempool (since they are now considered *confirmed*), and broadcasts the new block to other nodes in the network.

As shown, each node acts as a sentinel for the entire blockchain. This is why it is said that Bitcoin is fully *decentralized*, with no single point of failure. To "ban" Bitcoin, every node worldwide would have to be shut down. Such an action is impossible because Bitcoin nodes usually run on private networks that prevent them from being localized. The only way to stop Bitcoin is to stop the Internet altogether.

4.2 Bitcoin wallets: holding the keys

A *Bitcoin wallet* is best understood as a digital key-chain, not a purse. The software carries out three tasks on the user's behalf:

1. **Safekeeping of credentials.** At first launch, the wallet either imports or generates a random 256-bit *seed*, usually shown to the user as a 12- or 24-word phrase. From that single piece of entropy, it deterministically derives a long list of private–public key pairs. Whoever controls the seed controls every derived address; loss of the seed removes the corresponding bitcoins from circulation *forever*.

2. **Balance discovery.** The wallet asks one or more full nodes which of its addresses still hold unspent outputs (called UTXOs). Adding those outputs gives the owner's spendable balance—no central account ledger is consulted, as the check is performed directly against a valid copy of the public blockchain maintained by a Bitcoin node.

3. **Payment initiation.** To pay, the wallet selects sufficient UTXOs, builds a transaction, signs each input with the relevant private key, and broadcasts the signed message to the network's *mempool*. The signing keys themselves never leave the device.

Two economic implications follow. First, *ownership* in Bitcoin is purely cryptographic: whoever can sign with a key exercises full disposal rights, and there is no external register of names. Second, custody choices mirror familiar financial trade-offs. Users may keep the seed offline in a hardware device (maximising security but reducing convenience), deposit it with a custodial exchange (gaining ease of use at the cost of counterparty risk), or adopt hybrid solutions such as multi-signature schemes.

Because bitcoins themselves never leave the global ledger, a wallet's role is analogous to online banking software rather than a physical billfold: it is an interface to property rights, not a container for coins. The next subsection explains how the network enforces those rights at scale.

4.3 Bitcoin miners

There exists a third type of computer in the Bitcoin network, called *miners*. Miners are specialized computers that gather pending transactions from the mempool and assemble transaction blocks from them. To do so, miners should also solve a challenging puzzle called "Proof-of-Work". If they succeed, they propagate the new block to Bitcoin nodes, which will validate it and add it to their copies of the blockchain. Miners receive some bitcoins as a reward for doing so.

The mining process is key to understanding the strengths of the Bitcoin network. The following section provides a detailed description of their work.

5 Clearing Bitcoin Transactions: The Mining Process

In this section, we will see how transactions are settled in the Bitcoin network. This task is carried out by a large number of computers scattered around the globe, called *miners*. Its name comes from the fact that some lucky miners receive some bitcoins as a reward for their effort, just as a gold digger finds a gold nugget. As we will see, luck plays an essential role in this process.

A miner initiates the mining process by selecting pending transactions to clear, which, as stated above, are stored in a data structure called the *mempool*. Miners select some of them and group them in a so-called *transactions block*. Once the block is ready, before propagating it to the Bitcoin network, the miner should solve a challenging puzzle called *Proof-of-Work* (PoW) to gain the right to add the new transaction block on top of the blockchain.

Why is solving such a puzzle needed? Any miner could build a block of transactions in a fraction of a second. If the Bitcoin network allowed miners to propagate a new block as soon as they were ready, they would flood the entire network with blocks that would likely contain the same unconfirmed transactions. Therefore, there is a need for a mechanism that, first and foremost, establishes a slower pace for this task. This mechanism, called the Proof-of-Work, is a cornerstone of the Bitcoin network.

Bitcoin specifications stipulate that new blocks should be added approximately every 10 minutes. To ensure this, miners must solve the Proof-of-Work, a problem so complex that, on average, one single miner (out of tens of thousands worldwide) solves it every ten minutes. In other words, the problem's difficulty is set so that the combined computing power of all miners worldwide solves it every 10 minutes on average. The miner who solves this puzzle adds its block to the top of the blockchain, thereby clearing the associated transactions, and receives some bitcoins as a reward. As the new block propagates worldwide, its transactions are considered settled, and miners restart the process to settle other transactions still pending in the mempool.

To generate a valid block, miners follow a specific sequence of steps. A sketch of the general algorithm is presented in Appendix C. The process is carried out in such a way that the result is considered *immutable*, in the sense that it is impossible to change a settled transaction without being noticed.

The reward for successfully creating a valid block includes not only newly minted bitcoins but also all transaction fees associated with the settled transactions. This is why miners tend to prioritize transactions with higher fees for inclusion in the new block: Transactions with fees below the current mempool average risk being postponed indefinitely.

6 Why the Bitcoin blockchain is immutable

In the Bitcoin protocol, signed transactions can not be modified in any way[1]. A malicious actor cannot rebuild the transaction without the private keys of the inputs. The reason is that any changes to this transaction would drastically change the transaction hash, and therefore, the signatures would not match. More details on how this is ensured are provided in Appendix A.

The Bitcoin protocol extensively uses hashing to prevent changes to the transaction blocks in the blockchain. As shown in Fig. 1, which illustrates the Bitcoin blockchain structure, each new block contains the hash of the previous block. Hashes are the "signatures" that prevent blocks from being modified. Several security measures, described below, are in place to prevent malicious changes.

[1] Appendix A.2 describes in detail the signing transaction process.

1. Once a transaction T_x is signed with asymmetric encryption (see Appendix B), it can not be changed. This ensures that every transaction is recorded in the blockchain.

2. A malicious actor might like to add or delete a transaction from any block that is already part of the blockchain. To avoid this, when a miner builds a transaction block 1, it applies a special hash function to all the included transactions, generating the so-called *Merkle Root* hash of those transactions and storing it in the associated block header. See Fig. 1(1). Any addition or deletion of a transaction inside a block will invalidate this Merkle Root hash.

3. Since the Merkle Root hash changes when a transaction is added or deleted, our malicious actor might attempt to recalculate the Merkle root of the modified block, replacing this value in its block header. To avoid it, each block header includes *the hash of the header of the previous block*. See Fig. 1(2). This avoids any change in a block, since its new hash would not match the copy stored in the block upon it.

4. Finally, our malicious actor might decide to recalculate *the hashes of all the block headers* of the entire blockchain. Since hash calculation is relatively easy, this recalculation is perfectly possible. However, the miner who built each block managed to solve the Proof-of-Work puzzle associated with that block, and store the solution (called *nonce*) in its block header. See Fig. 1(2). Recall that solving the Proof-of-Work is a kind of competition that is won by one single miner around the globe every ten minutes on average. Our malicious actor cannot find the new nonce required to solve the Proof-of-Work associated with any modified block in a reasonable amount of time, let alone finding *all the nonces* of the blocks upon it.

These characteristics make it mathematically impossible to modify any data in the blockchain. The Bitcoin community acknowledges that once a block is confirmed in the blockchain and five new blocks are stacked on it (that is, six *confirmations*), the transactions in that block can be considered immutable.

A superficial intuition might suggest that an inconspicuous alteration to a data structure spanning several hundred gigabytes could pass undetected. In practice, each node that downloads the blockchain performs a comprehensive integrity check before accepting it, as detailed in the next section.

7 Trusting a downloaded copy of the blockchain

As we saw in the previous section, the Bitcoin blockchain is designed so that it is impossible to change any single datum on it without being noticed. Any interested user can set up a Bitcoin node at home. Once the user installs the Bitcoin software, the next step is to download a copy of the Bitcoin blockchain. How can this user *trust* that the downloaded copy has not been manipulated?

The answer is simple. The Bitcoin software does not *trust* that the downloaded blocks are correct, but *verifies* them. To get a copy of the blockchain, our Bitcoin software will take the following steps:

1. The Bitcoin software that runs in the new node includes, in its code, a copy of block 0 of the Bitcoin blockchain, called the *Genesis block*). The node starts by calculating the hash of the block header of block 0.

2. The node requests the following block (i.e., block 1) from other Bitcoin nodes connected to the Bitcoin network. When it receives a copy of that block (remember, a copy sent by an untrusted source), our node compares the hash value calculated in the previous step with the hash value stored in the block header of block 1.

3. If both values match, our node checks the correctness of the entire block 1: It checks the signatures of all transactions included in that block; calculates the Merkle root hash of all transactions in that block and compares it with the value stored in its block header; and verifies that the nonce of block 1 effectively solves the Proof-of-Work puzzle for that block.

4. Once our software acknowledges block 1 as valid, it adds block 1 to its copy of the blockchain and requests block 2. The process is repeated for all the blocks that form the blockchain until the entire downloaded copy has been verified.

Note that our computer will not simply "trust" that the downloaded blockchain is correct, but *verifies every single block of transactions*. When the process finishes, the Bitcoin software ensures the entire blockchain is valid.

8 The Infamous Proof-of-Work

In the previous sections, we stated that, to produce a valid block, miners should solve a complex puzzle called Proof-of-Work. We will now briefly describe what kind of puzzle it is. To do so, we should introduce the concept of *hash functions*. A hash function is a mathematical function that, for a string of bytes of arbitrary size, returns a numerical value of fixed size, that acts as a "digest" of that input[2]. Hash functions have several properties that are critical for the proper functioning of the Proof-of-Work mechanism.

As we saw previously, the idea behind Proof-of-Work is to set up a problem so difficult that the entire computing power worldwide devoted to solving it only arrives at a solution every 10 minutes on average. One might expect that the problem would be highly complex to state. However, the problem is very easy to describe. Let BH be a block header. As we saw previously, a block header contains various information, including the hash of the previous block header and the Merkle Tree value of the transactions included in that block. The block header also includes a numerical value n, called *nonce*. If we apply a specific hash function to BH, we will obtain a hash h. The puzzle involves finding a value for the nonce n that yields a value of h below a specific value, known as the *difficulty threshold*.

Hash functions, as described in detail in Appendix B, possess several properties essential to the Bitcoin protocol. The first property that hash functions should fulfill is to be *one-way operations*: If we need to find a nonce that leads to a specific value for the hash, it is essential that this nonce cannot be derived from the hash value. Therefore, to find a hash lower than n, all we can do is to try different values for the nonce, calculating the hash of the block header containing it, and comparing that hash value with the threshold n, until one nonce makes the hash smaller than this threshold.

Other properties of hash functions are equally important. Hash functions should be *deterministic* because the hash value should always be the same for a given block header and nonce. This allows any node to verify that the hash value of a given block header meets its threshold requirements. Hash functions should also be *computationally efficient*: Since finding the nonce that fulfills the condition requires trying many nonces, this task should be easy to calculate. It is also important that hash functions present *avalanche effect*, because this ensures that changing the nonce slightly, let's say from 22 to 23, will produce an entirely different hash value. Finally, hash functions should present an *efficient distribution*, that is, the generated hashes should be uniformly distributed across the solution space. This helps establish the problem's difficulty properly: If we cut the threshold in half, we expect the problem to be twice as difficult to solve, and this is only true if the hashes are uniformly distributed across the solution space.

Now we can understand why the Bitcoin network is considered immutable. The entire purpose of the Proof-of-Work mechanism is to make it impossible to modify the blockchain. Recall Fig. 1(2): Each A block header includes the Merkle root of all the transactions in the block, the hash of the previous block header, and a nonce that makes the hash of this block header below a certain value. The "avalanche effect", described above, means that any slight modification in any of these transactions will drastically change the Merkle root value, so the block header's hash will no longer be below the threshold. To fix it, we would need to find a new nonce. However, to find the old one, the combined computing power of the Bitcoin network worldwide needed around ten minutes. The probability of finding a new nonce with a single computer in any reasonable amount of time is approximately zero. Even worse, we will need to find a new nonce for the modified block. As all block headers contains the hash of the previous block, and that the hash of our block header will change when we change the nonce, to keep the consistency of the blockchain, we would need to *calculate sequentially the new nonces for all the blocks on top our modified block*, a task impossible to carry out in any reasonable time.

[2]Appendix B.1 describes hash functions and enumerates their most essential properties.

8.1 Adjusting the difficulty of the Proof-of-Work

As we saw in the previous section, the Proof-of-Work mechanism consists of finding a nonce value below a certain threshold. The threshold is set so that the combined mining power of the Bitcoin network reaches a solution every 10 minutes. However, this combined computing power is not constant. On the one hand, many new miners are deployed daily, and new, more powerful miners enter the market periodically. On the other hand, many miners are also shut down daily worldwide, due to rising energy prices or bans on mining activity in certain countries for geopolitical reasons.

How does the Bitcoin network automatically set the difficulty of the Proof-of-Work mechanism? The Bitcoin protocol establishes a clever mechanism to adjust the difficulty of the problem. Approximately every two weeks (more precisely, after adding 2016 new blocks to the blockchain), all Bitcoin nodes simultaneously calculate the time it took to mine those blocks and adjust the difficulty threshold accordingly to keep the average block time close to 10 minutes. If this average time were precisely 10 minutes, the threshold n would not need any change. If the average time between consecutive blocks was, let's say, 10'30", that is, 5% above the desired value, it means that the mining process took more than expected, and that there is likely less aggregated computing power working in the Bitcoin network. Therefore, to compensate for this reduction in the network's accumulated computing power, the problem should be 5% easier; the new threshold is set to the old one multiplied by 1.05. In other words, if the old value of n were 100 (that is, miners were trying to find a hash whose value was lower than 100), the new threshold would be 105, making miners' lives somewhat easier. From that moment on, all Bitcoin nodes will verify that all new blocks proposed fulfill this new condition before adding them to their local blockchain copies. They do not need to reach consensus on the value of the new threshold: they arrive at the same conclusion concurrently.

It is important to highlight that this *difficulty adjustment* is not a centralized decision. The difficulty adjustment procedure is a rule of the Bitcoin proposal, and therefore, all nodes follow it simultaneously, arriving at the same threshold value. If a rogue node decides to accept blocks that do not meet these criteria, nobody else will accept any block redistributed by it.

8.2 How the Proof-of-Work Helps to Secure the Bitcoin Network

The Proof-of-Work (PoW) mechanism is indeed energy-intensive, but this energy expenditure is essential to Bitcoin's security and decentralization. We will enumerate four reasons why Proof-of-work (PoW) is considered the most effective way to secure the Bitcoin network.

Suppose the Bitcoin network consists of ten miners. An attacker could deploy 15 additional miners that follow a different set of Bitcoin rules, leaving the original miners in the minority. With PoW, adding more miners does not ensure control of the network. Since mining requires significant energy, PoW ensures that control over the network cannot be achieved by simply deploying new miners. Instead, miners must commit energy, a scarce resource, to participate. The larger the Bitcoin network, the more miners and energy should be committed.

Second, as we saw previously, Bitcoin's PoW mechanism makes it practically impossible for any single entity to rewrite the blockchain, thus maintaining a secure, tamper-resistant ledger. Other consensus mechanisms, such as Proof-of-Stake (PoS), in which nodes that stake the most coins are entitled to add new blocks to the blockchain, are more vulnerable to attacks, as entities with substantial ownership could theoretically control the network. We will examine PoS in detail in the following section.

Third, PoW rewards miners with new bitcoins and transaction fees, aligning incentives by ensuring they benefit financially only if they play by the rules. If miners tried to act dishonestly or override network rules, they would lose substantial investments in energy and equipment without any reward.

Finally, with PoW, anyone with sufficient computing power and access to electricity can participate in mining, thereby promoting decentralization. In contrast, PoS centralizes mining around nodes with substantial coin holdings. PoW creates a more balanced playing field, where participation depends on computational power rather than ownership of coins.

In summary, PoW secures Bitcoin by creating a system where participants need to commit real-world resources, making attacks economically unfeasible and ensuring consensus and decentralization. While energy-intensive, this mechanism underpins the integrity and security of the network in ways that alternative mechanisms have yet to match.

8.3 Proof-of-Stake as an Alternative to Proof-of-Work

Several cryptocurrencies that emerged after Bitcoin have replaced the Proof-of-Work mechanism with other methods that require far less energy, at the cost of centralizing mining. One widely used alternative is the "Proof-of-Stake" (PoS), currently used by Ethereum. Unlike Proof-of-Work, which relies on computational power to solve a cryptographic puzzle, PoS selects validators based on the amount of cryptocurrency they "stake" or lock up as collateral. A selected validator proposes a new block, and a committee of other validators verifies its correctness. Validators cast their "votes" to either approve or reject the proposed block. Ultimately, validators receive rewards for proposing valid blocks and contributing to the validation process. In this way, entities with a higher stake in a particular cryptocurrency are incentivized to maintain honest behavior, as malicious actions can result in the loss of some or all of the staked cryptocurrency associated with this blockchain. Therefore, PoS does not require spending a massive amount of energy [11]. Instead, it relies on *trusted* validators.

The proof-of-stake (PoS) mechanism used in Ethereum exhibits some interesting properties. The average time required to settle a transaction is lower than when using Proof-of-Work (PoW). In addition, the difficulty adjustment associated with a sudden change in the incoming transaction rate is more flexible. The Bitcoin difficulty adjustment occurs every two weeks. On the other side, Ethereum maintains block times of around 12 seconds by adjusting validator incentives and penalties to align behavior rather than using difficulty levels. Finally, Ethereum transactions include a base fee automatically calculated by the protocol based on network congestion, which is "burned" (i.e., permanently removed from circulation). This process effectively reduces the total supply of its token (called *ether*) over time, introducing a deflationary mechanism.

However, PoS mechanisms have drawbacks. Unlike PoW, where a malicious actor must control at least 51% of the mining power to compromise the blockchain (an impossible task nowadays due to the distributed nature of mining), in PoS, control over the network depends on owning a significant share of the cryptocurrency being staked. Acquiring a majority stake may be easier for a wealthy actor simply by buying enough coins.

Besides this, to carry out a 51% attack in PoW requires expensive, dedicated mining machines that cannot be easily used for other purposes afterward. In the case of PoS, no expensive mining facilities or energy are required, but only a certain amount of the corresponding cryptocurrency is needed. An attacker who gains the majority of the stake and subsequently attacks the blockchain could potentially sell their tokens after the attack, recouping a significant portion of their costs.

Finally, PoW inherently resists "long-range attacks" (where an attacker rewrites the blockchain from a distant past) because rewriting the chain requires redoing all the computational work, which is infeasible. By contrast, PoS is more vulnerable to such attacks because validators must only resign old blocks, which costs them nothing because they are entitled to do so. Special mechanisms, such as "finality" checkpoints or slashing penalties, are necessary to mitigate this risk; however, these mechanisms introduce complexity and rely on social coordination to enforce honest behavior [9].

In summary, Proof-of-Stake is a more energy-efficient solution than Proof-of-Work; however, precisely because of this, it is more challenging to establish a robust defense against network attacks.

9 Bitcoin Issuance

In the first sections of this work, we briefly stated that Bitcoin issuance is fixed and cannot be changed. Now that we understand how all Bitcoin nodes simultaneously carry out Proof-of-Work difficulty adjustment, we will immediately see why Bitcoin issuance is enforced in the Bitcoin protocol.

As we saw in Section 5, miners that successfully mine a block receive some bitcoins as a reward. These bitcoins come from two sources. Part of them are the fees included in each transaction in the mined block, and some are minted in a special transaction, called *Coinbase transaction*, that the miner inserts as the first transaction of the block. The Coinbase transaction has no inputs and one output, and assigns a fixed number of bitcoins to that output (which belongs to the miner).

The Bitcoin protocol includes the following two rules related to the issuance of new bitcoins in the Coinbase transactions:

1. The first 210,000 blocks of the Bitcoin blockchain should include 50 new bitcoins each in their Coinbase transaction.

2. Every 210,000 blocks (approximately every four years), the reward should be cut in half.

These four years are called *epoch*, and the event of cutting the reward by half is called *halving*. Therefore, the reward during the first epoch was 50 bitcoins. After the first halving, the reward was reduced to 25 bitcoins; after the second halving, to 12,5 bitcoins, and so on. These two rules make the Bitcoin issuance (S) follow this geometric series:

$$S = a \sum_{n=0}^{\infty} 210,000 \times \frac{50}{2^n}$$

Using the formula for the sum of an infinite geometric series, that is:

$$\sum_{n=0}^{\infty} r^n = \frac{1}{1-r}, \text{where } r = \frac{1}{2},$$

we have that

$$S = 210,000 \times 50 \times \frac{1}{1-\frac{1}{2}} = 21,000,000$$

These 21 million coins are the maximum cap for Bitcoin issuance. Since bitcoins have eight decimal positions, they are not infinitely divisible. Therefore, the last epoch will have a reward per mined block of just 0.0000 0001 bitcoins, or 1 *satoshi*. When that epoch terminates (around 2140), no more new bitcoins will be issued, and miners will rely solely on transaction fees to support their activity.

As Bitcoin's market value continues to rise, concerns arise about whether it will remain usable for small everyday payments. Since bitcoins are divisible only up to eight decimal places, the smallest unit (one satoshi) could become too valuable to facilitate microtransactions if Bitcoin's price appreciates significantly. In such a scenario, layer-2 solutions like the Lightning Network [10] will play a central role in enabling high-frequency, low-value payments by aggregating and settling them off-chain. These second-layer mechanisms are designed to ensure that, despite the indivisibility of satoshis, Bitcoin remains functional as a medium of exchange even at high valuations.

The reader may think that, since a miner builds the transaction before finding a nonce that solves the PoW puzzle, it can put an arbitrary number of bitcoins in the Coinbase transaction. However, this is not the case because the block would not be valid. In the same way that the automatic difficulty adjustment is carried out by all Bitcoin nodes simultaneously, all of them carefully check any new transaction block proposed to be added to the top of the blockchain. Any transaction block that includes an invalid number of bitcoins in its Coinbase transaction will be immediately rejected by all honest nodes connected to the Bitcoin network and will not be included in the blockchain. This explains why Bitcoin issuance is immutable.

It is clear that each halving event has significant consequences for Bitcoin miners' profitability. Keep in mind that, as with gold mining, Bitcoin mining is probabilistic: In the long term, miners aim to earn enough bitcoins to cover their costs. Energy cost is not affected by a halving event, but the profitability of the Bitcoin mining industry is. Older Bitcoin mining equipment, which barely broke even during a given epoch, becomes unprofitable after a halving by a noticeable margin. Consequently, the competitive mining industry expels the less efficient miners. This is why the global hashrate decreases shortly after a halving event, and therefore, the elapsed time between new blocks tends to increase. The difficulty adjustment comes to the rescue, setting new threshold values that gradually stabilize the situation.

9.1 Halving Events and their Impact on Bitcoin's Price

Bitcoin's halving events, occurring approximately every 4 years, reduce the block reward for miners by 50%, effectively decreasing the new bitcoin issuance rate. This programmed supply shock has historically been associated with price appreciation due to its impact on the supply-demand equilibrium.

Each halving event reinforces Bitcoin's scarcity, aligning it more closely with deflationary assets, such as gold. As the available supply diminishes, speculative and institutional interest tends to increase, contributing to price surges in previous cycles. However, while historical trends suggest a correlation between halvings and price rises, causation remains debated, with other macroeconomic factors playing significant roles in Bitcoin's valuation [3, 6].

Market efficiency suggests that the impact of halvings should be priced in beforehand, yet past market cycles indicate post-halving price rallies [2]. These movements are likely driven by shifts in investor sentiment, increasing adoption, and the reinforcing narrative of Bitcoin as digital gold. Over time, as Bitcoin's issuance rate approaches zero, the reliance on transaction fees for miner incentives will become a critical factor in sustaining network security and stability [7].

Bitcoin's transparent and deterministic issuance schedule enables the development of predictive econometric and simulation models. These models can serve as decision-making tools to analyze potential inflationary and deflationary trends, stress-test monetary policy alternatives, and evaluate macroeconomic responses to Bitcoin integration. The predictability of Bitcoin issuance and halving events supports the integration of blockchain-based instruments into long-term macroeconomic decision support systems (DSS) [8].

10 Conclusions

This technical companion provided economists with a first-principles account of how Bitcoin secures ownership, validates transactions, and coordinates a single ledger history without trusted intermediaries. We emphasized the protocol mechanisms that generate economically salient properties such as censorship resistance, credible scarcity, and probabilistic settlement finality via confirmations.

Bitcoin's transaction model ties spending authority to the ability to produce valid signatures under the script rules, shifting key elements of property rights and custody from institutional account ledgers to cryptographic control. Ledger integrity follows from hash-linked blocks and Merkle commitments, which make tampering increasingly infeasible as additional Proof-of-Work accumulates. Full nodes enforce the rules by independently verifying blocks and transactions, while miners provide the costly coordination mechanism that selects a canonical chain under latency and adversarial conditions.

The issuance schedule is therefore not a policy intention but a mechanically audited rule: nodes validate the coinbase and enforce the halving path, constraining discretionary expansion. Looking ahead, as block subsidies decline, security increasingly relies on fee revenue, bringing the design and economics of fee markets, scaling layers, and settlement demand to the foreground. A productive research agenda is to map these protocol primitives into standard economic objects, including payment choice under finality risk, custody and intermediation, and the industrial organization of mining under regulation and energy price shocks.

11 Acknowledgments

Prof. Llanos acknowledges financial support by the Spanish Ministerio de Ciencia e Innovación and by the European Regional Development Fund (ERDF) program of the European Union, under Grant PID2022-142292NB-I00 (NATASHA Project). Prof. Javier Perote acknowledges financial support by Castilla and León Regional Government (Spain) under project SA094P24. ChatGPT and Grammarly were used to gather suggestions that improve the readability of this work.

References

[1] A. M. Antonopoulos and D. A. Harding. *Mastering Bitcoin*. O'Reilly Media, Inc., 2023.

[2] D. El Mahdy. The economic effect of Bitcoin halving events on the us capital market. *Accounting and Finance Innovations*, 65, 2021.

[3] J. Fabus, I. Kremenova, N. Stalmasekova, and T. Kvasnicova-Galovicova. An empirical examination of Bitcoin's halving effects: Assessing cryptocurrency sustainability within the landscape of financial technologies. *Journal of Risk and Financial Management*, 17(6):229, 2024.

[4] P. Gauravaram. Cryptographic hash functions: Cryptanalysis, design and applications, 2007. Queensland University of Technology.

[5] D. Hankerson and A. Menezes. Elliptic curve cryptography. In *Encyclopedia of Cryptography, Security and Privacy*, pages 1–2. Springer, 2021.

[6] I. Jiménez, A. Mora-Valencia, and J. Perote. Bitcoin halving and the integration of cryptocurrency and forex markets: An analysis of the higher-order moment spillovers. *International Review of Economics & Finance*, 92:302–315, 2024.

[7] D. Lasi and L. Saul. A system dynamics model of Bitcoin: Mining as an efficient market and the possibility of "Peak Hash". *arXiv preprint arXiv:2004.09212*, 2020.

[8] G. M. Marakas. *Decision Support Systems in the 21st Century*. Prentice Hall, 2003.

[9] U. Pavloff, Y. Amoussou-Guenou, and S. Tucci-Piergiovanni. Ethereum proof-of-stake under scrutiny. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, pages 212–221, 2023.

[10] M. Qi, Q. Wang, Z. Wang, M. Schneider, T. Zhu, S. Chen, W. Knottenbelt, and T. Hardjono. SoK: Bitcoin Layer Two (L2). *arXiv preprint arXiv:2409.02650*, 2024.

[11] C. Stoll, L. Klaaßen, and U. Gallersdörfer. The carbon footprint of Bitcoin. *Joule*, 3(7):1647–1661, 2019.

A A Didactic Approach to Bitcoin Transactions

We will explain what Bitcoin transactions are and how they are settled in the Bitcoin network. To introduce the basic concepts (addresses and transactions), this section will start with a simple analogy that illustrates the technological solutions involved. Next, we will discuss the mathematical tools that enable the implementation of the Bitcoin transaction system and how they support the Bitcoin protocol.

For now, let's set aside the fact that Bitcoin operates on a distributed ledger system. To understand how transactions are settled in the Bitcoin network, their nature, and some of their peculiarities, we will consider Bitcoin a sort of bank where different users have accounts denominated in bitcoins. For reasons that will be explained later, each user typically holds multiple Bitcoin accounts rather than just one. Each account is referred to as an *address*, and each address is associated with zero or more bitcoins at a given moment.

Suppose that Alice has exactly 0.002 bitcoins in one single address at the Bitcoin bank, and she would like to send all of them to Bob in exchange for a particular service. To do so, Alice would issue a *transaction* to the bank. The transaction will contain her address as input, the amount, Bob's address as output, and her signature. As shown in Fig. 2, this transaction is somewhat similar to a check.

However, there is a slight difference. In a real-world check, Alice does not need to disclose the amount of money stored in her input address. She would state her account, indicate that Bob should receive a specific amount, and have the institution debit her account for that amount. In the Bitcoin protocol, however, transactions differ slightly. In a Bitcoin transaction, all the bitcoins in an input address are spent *entirely* by transferring them to a *destination* address, and to a *change* address that belongs to the issuer.

In Fig. 2, the amount to be transferred matches precisely with the bitcoins in Alice's input address. A more realistic situation is depicted in Fig. 3. In this case, Alice's address 1 is spent entirely: part of her bitcoins goes to Bob's address 1, and the remaining part goes to another address owned by Alice.

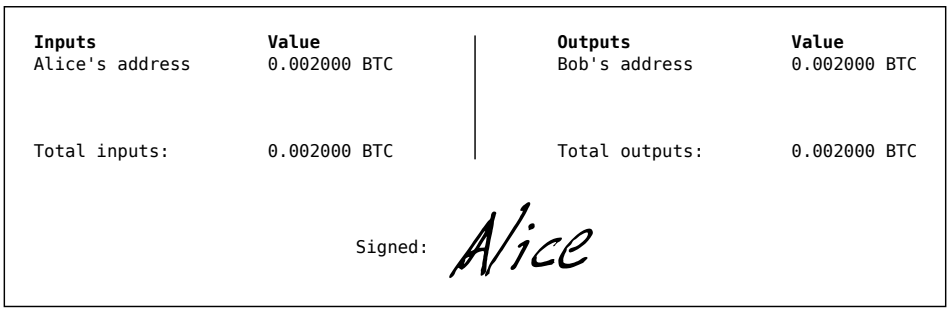

Figure 2: A straightforward transaction in our Bitcoin bank

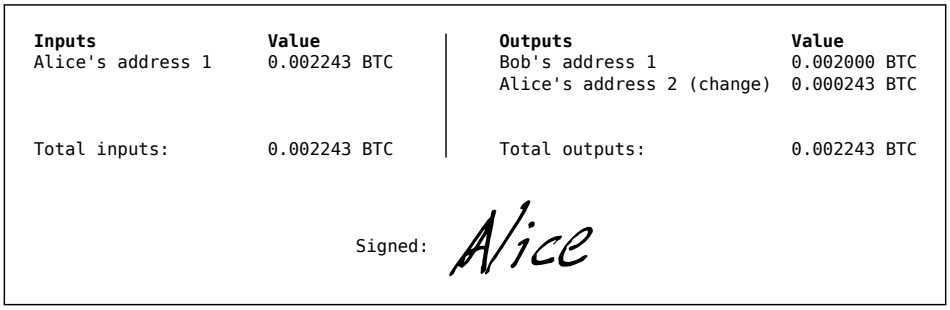

Figure 3: This transaction includes Bob's destination address and Alice's change address.

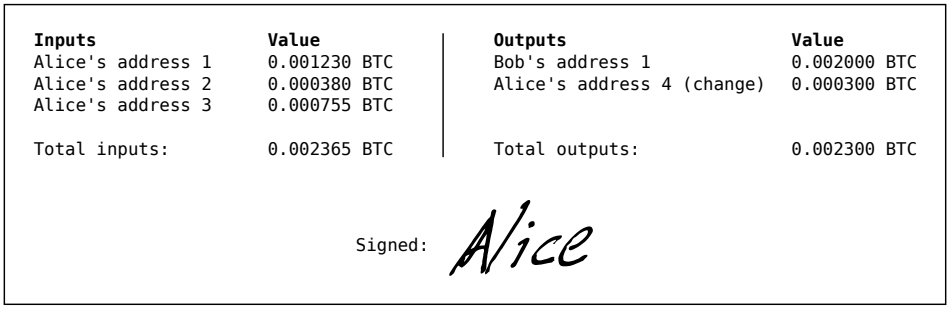

Figure 4: A more realistic transaction, with several inputs, several outputs, and an implied fee.

This is why Bitcoin users manage many addresses instead of just one. As we will see later, as long as the Bitcoin transaction history (the ledger) is public, it is not recommended to reuse addresses, to avoid a third party from relating all the user's transactions.

What happens if Alice does not have enough bitcoins in her addresses to pay for Bob's services? In this case, the transaction should include *several* input addresses. See Fig. 4. Besides this, the reader can observe that the inputs in this third version of our transaction do not exactly match the outputs. The difference is the *fee* collected by the entity responsible for clearing transactions in the Bitcoin system. Fees in Bitcoin work slightly differently from those in the real world. First, fees are not proportional to the amount transferred: in our example, the 0.000065 fee would work equally well for a 0.002243 BTC transaction as for a 100,000-BTC transaction. Second, the issuer of the transaction freely chooses the fee offered. As we will see later, the entities that process incoming transactions tend to process those that provide higher fees first. Therefore, to speed up transaction settlement, the issuer may offer a fee higher than the average fee for other unsettled transactions. As a consequence, fees fluctuate continuously in the Bitcoin network.

Since all the bitcoins from each input address are spent entirely, there is no need to list their value in the transaction: The system in charge of validating and storing the transaction (called *miner*) will first check how many bitcoins remains in each input address, and whether the sum of the amounts are enough to cover the amounts of the outputs, taking the difference (if any) as a transaction fee.

This analogy is sufficient to understand several key concepts of the Bitcoin network, including addresses, transactions, and fees. However, other elements should also be translated to the digital world. For example, suppose that Craig knows Alice's input addresses. What prevents Craig from issuing a transaction that transfers Alice's bitcoins to his address? The answer is: Alice's signature. All transactions are signed by the issuer. We will briefly review the technology behind digital signatures in the following paragraphs.

A.1 Bitcoin addresses

Asymmetric encryption and digital signatures are the cornerstone of the Bitcoin network. As we have seen above, both concepts are intrinsically related: a digital signature uses a private key to sign the statement, and the corresponding public key can be used to verify the signature. We need to know the

basics of these cryptography techniques because their application is key to demonstrating ownership of assets in a distributed, decentralized system like the Bitcoin network.

Once we have described asymmetric encryption and its use in signing messages, we will explain its role in Bitcoin transactions. In the Bitcoin network, each user manages multiple pairs of public/private keys. *Each public key is used as a Bitcoin address, which can be associated with funds, while the associated private key is needed to spend them.* Key pairs are generated by an algorithm that uses a random number, called a *seed*, which the owner should keep secret. The entire process of generating a seed, deriving public and private key pairs, and using them to interact with the Bitcoin network is carried out by special applications called *Bitcoin wallets*. The user is only responsible for keeping a physical copy of the seed in a *very* secure place. If the user loses the key, his/her bitcoins are lost forever.

Suppose Bob wants to receive some bitcoins from Alice but does not have a Bitcoin address. Bob should generate a set of suitable private-public key pairs for use on the Bitcoin network. To be completely secure, this generation should start by choosing an extremely large random number, called a seed. As a technical note, Bitcoin uses the `secp256k1` Elliptic Curve Cryptography (ECC) standard [5] to generate a set of private-public keys from a given seed. The description of this algorithm is beyond the goals of this work. However, it is critical to understand the importance of the seed in this process.

To generate a set of Bitcoin private-public key pairs, we will need a single random number between 1 and 1.1578×10^{77}, a value that can be codified as a 256-bit number. If the seed isn't *truly random*, it could be vulnerable to a brute-force attack, where someone tries different combinations with the help of a computer, until they find the correct one. In fact, an extremely secure way to generate the seed is to flip a coin 256 times, record the outcomes, and check whether the resulting number is less than or equal to the upper limit above. If this condition is met, the seed is appropriate. Although this method can be effectively used, nowadays, any computer or smartphone can generate seeds at the required level of randomness.

Once a random seed has been chosen (and its physical copy is stored elsewhere), Bob will use it to generate pairs of private-public keys using the `secp256k1` algorithm. A public key, which will be later used as a Bitcoin address, might have the following aspect:

bc1qz9zznnpkjfdjx2r903rn86f3rkjamue8kf5s0w

As long as Bob is the owner of the seed that generated his particular public/private key pair, Bob is the *owner* of this address. Unlike the traditional banking system, where ownership of a given account is stored in a centralized repository, and which can be demonstrated by presenting an ID card, in Bitcoin, whoever has the private key associated with a Bitcoin address is effectively the owner of that address. Therefore, as the owner, Bob can transfer these bitcoins to Charlie if he wants to. Nobody can obtain or use Bob's private key since a private key cannot be derived from a public key. This is both an advantage and a drawback. On the one hand, nobody will ever spend the bitcoins associated with Bob's address without his private key (or the seed used to generate both). On the other hand, if Bob loses the seed, the bitcoins associated with it *are lost forever*.

As we mentioned earlier, this concept of ownership is radically new because it is the first time a good with economic value is permanently linked to a piece of information. A physical good can be locked in a vault using a key combination, but vaults can be broken into. Any physical good of economic value can be stolen or seized by a government. However, Bitcoin public/private keys are mathematically secured and are unbreakable, and seeds can be memorized (for better or worse). This concept of ownership challenges existing legal frameworks governing ownership and wealth distribution.

We can now revisit Fig. 4, replacing Alice's and Bob's addresses with Bitcoin addresses. The result is shown in Fig. 5. Note that Alice's and Bob's names are no longer needed in the transaction. As long as the Bitcoin ledger is public, anyone can view this transaction, but it is impossible to determine the owners' names. This is how the Bitcoin system ensures privacy: No personal information is stored in any transaction. We can also remove the input values, because, as we said above, the miner responsible for validating the transaction will recover their current values from the transaction history.

Although the identities are removed, Bob, as the owner of the first output address, can link the input addresses as belonging to Alice. Moreover, Bob can check how many bitcoins were associated with any of them before the transaction. This poses a significant privacy risk: Suppose that Alice bought 500 bitcoins for $500 in 2011 and stored her 500 bitcoins in a single Bitcoin address. If she uses this address to pay for one coffee, Bob can see that the change will be almost 500 bitcoins, which

Inputs		Outputs	Value
1ChsiTeX...ujLY7zEr		bc1qz9zz...e8kf5s0w	0.002000 BTC
7f445356...50d884c9		f8c17dfa...e7f21519	0.000300 BTC
bc1quhru...kq0r8l2d			
(Total inputs:	0.002365 BTC)	(Total outputs:	0.002300 BTC)

Figure 5: The transaction of Fig. 4, replacing Alice's and Bob's addresses with proper Bitcoin addresses (shortened for convenience). Note that nothing suggests the names of the owners of each address. Input values have also been removed because they will be retrieved from the transaction history. Miner will also calculate the total inputs and outputs, ensuring the former cover the latter, and use the difference as the transaction fee when settling the transaction.

represents around 50 million dollars at the time of writing. This is like paying for a coffee with a 50-million-dollar bill, fully disclosing Alice's wealth. Unlike a check, which does not show the issuer's balance, Bitcoin transactions are fully traceable. This is why Bitcoin users are advised to distribute their balance in many addresses, governed by the same seed, to avoid exposing their wealth to others. Nowadays, it is commonly assumed that a convenient amount to store at each address is around 0.01 bitcoins. This number is subject to change as Bitcoin will likely appreciate.

A.2 Signing Bitcoin transactions

Now comes a critical part of our description of Bitcoin transactions and addresses. How can Alice *demonstrate* that she owns the three input addresses of the transaction represented in Fig. 5, and therefore has the right to spend them?

As stated above, the ownership of a particular address in Bitcoin (remember, a public key) is demonstrated by showing that the user possesses the corresponding private key. The following procedure should be applied to sign the transaction:

1. Alice applies a hash function to the entire transaction, obtaining a hash value. Bitcoin uses a hash function called "double SHA-256". This hash function produces a 32-digit hexadecimal hash. See Fig. 6(a).

2. Remember that each input address corresponds to a public key, whose corresponding private key is owned by Alice. For each input address, Alice constructs a text string composed of the input address and the hash of the *entire* transaction. Then she applies its associated private key to encrypt this string, effectively generating a signature, and stores each signature beside its corresponding input. In our example, Alice should repeat the signing process three times since there are three inputs. See Fig. 6(b).

The result is a digitally signed transaction. How can we check that the owner of the inputs has signed the transaction? The verification is not difficult. The following steps should be carried out:

1. We should first get the original transaction. To do so, we take the signed transaction depicted in Fig. 6(b) and strip off the transaction hashes and signatures. We will obtain the unsigned transaction shown in Fig. 6(a).

2. We apply the hash function to the stripped transaction. In our example, we will obtain the value `a1e6...6bdc`. As long as this hash matches the hashes included in the signed transaction, this demonstrates that the transaction has not been manipulated, for example, by changing an output address.

3. Once we have checked that the transaction hash is correct, we decrypt each signature with the corresponding public key (that, remember, is simply the input address!). By doing so, we will obtain a text string composed of the address and the transaction hash, identical to the information associated with each input.

The verification process is then complete: Whoever has signed the transaction holds the three private keys associated with the three input addresses, and therefore has the right to spend those

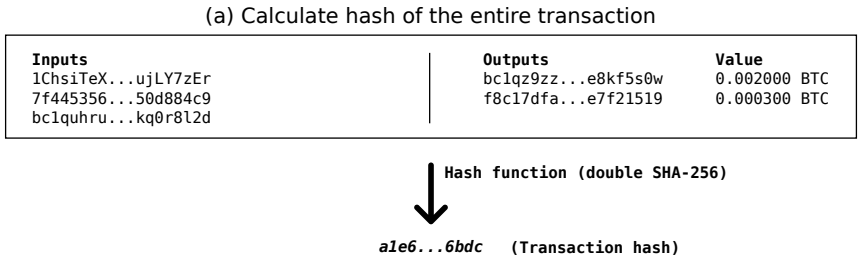

Figure 6: Signing process of the transaction depicted in Fig. 5. First, a hash function is applied (a), and then the hash is attached to each input address, signing all the information with its corresponding private key (b). At this point, the transaction is considered signed. Bitcoin addresses, the transaction hash, and the signatures have been shortened for convenience.

funds. Note that this simple definition of "ownership" eliminates the need for a central authority that officially states who the owner of each Bitcoin address is. This definitionhas profound implications, redefining the concept of private property.

The main consequence of the way transactions are built is that they are *immutable*. Once constructed, it is not possible to change any data. Suppose someone tries to change output addresses or bitcoin amounts. In that case, the transaction hash, calculated as shown in 6(a), will be completely different, and the signatures associated with the input addresses will not match them. This is why the "avalanche effect" property of the hashing function is so important: A slight change in the input leads to a completely different hash.

Regarding the security of the `secp256k1` cryptography method, let's put the size of the $[1, 1.1578 \times 10^{77}]$ random seed space into perspective. The number of atoms in the observable universe is around 10^{82}; one seed exists for every 10,000 atoms in the observable universe. Considering that there are currently around 55 million non-zero Bitcoin addresses stored in the blockchain, even if we could generate 1 billion keys per second and check whether each corresponds to a valid Bitcoin address, it would still take approximately 4×10^{54} years on average to guess a single private key. Therefore, we can conclude that the probability of guessing any `secp256k1` seed used so far is effectively zero.

A.3 Preserving the seed

The above paragraphs state the importance of preserving the random seed used to generate our set of public and private keys. Anyone with the seed is effectively the *owner* of the associated bitcoins. To ease the secure storage of the seed, there is a Bitcoin standard, BIP39, that converts any valid seed into a sequence of 12 or 24 words, called a *seed phrase*. These words are taken from a table of 2,000 words, and should be stored strictly in order. A possible 12-word seed phrase is the following (do not use it to generate keys!):

client regular museum craft crawl same abstract finger bag scorpion fiscal island

To secure our keys, we should store our seed phrase securely. Note that to memorize it exclusively is a bad idea: If something happens to us, our bitcoins are *lost forever*. Therefore, storing it in a place where our heirs can find it is essential. This is a consequence of making the ownership *so* private, and may act as a hurdle to the adoption of Bitcoin among individuals and institutions.

B Cryptography in a nutshell

To understand how transactions are digitally signed and why it is virtually impossible to falsify an issue's signature, we should first understand three key concepts: hash functions, asymmetric encryption, and digital signatures. This section is self-contained and introduces these techniques; no special background is necessary to understand it fully. As we will see in the remaining sections, these three concepts are used repeatedly, so it is essential to understand them properly to recognize Bitcoin's unique properties.

First, it is necessary to introduce some common computer science terms that the reader might already know. These are *bits*, *bytes*, *hexadecimal digits*, and *text strings*. These definitions, presented in a non-technical manner, will help us understand the foundations of Bitcoin technology.

Bit: A 1-digit numerical value of 0 (zero) or 1 (one). The name is a contraction of the term "Binary digiIT". Binary is a numbering system similar to decimal, but with base 2, that is, with only two digits. Like the decimal system, binary is also a positional system, where each digit has a value that depends on its relative position inside the number. Binary numbers are written as 0 (zero), 1 (one), 10 (two), 11 (three), etc.

Byte: An 8-bits binary value. The smallest natural number that fits in one byte is "0000 0000", that is, zero, and the biggest one is "1111 1111', that represents 255 in decimal.

Hexadecimal digits: Another numbering system, in this case with base 16. Therefore, it has 16 different digits, namely 0, 1, 2, ..., 8, 9, A, B, C, D, E, and F. It is widely used in computer science because four bits correspond exactly to one hexadecimal digit: 0000 in binary equals 0 in hexadecimal, and 1111 equals F. Therefore, one byte can be represented either by eight binary digits or by two hexadecimal digits.

Strings: Any arbitrary sequence of characters, such as "`This string has 30 characters!`". Note that inside a string we can have uppercase and lowercase letters, numbers, symbols, and even blank spaces (which are also valid characters). Since computers deal only with bytes, a table called ASCII assigns an 8-bit value to each Western character. For example, the character 'A' has the decimal value 65, which in binary is 0100 0001, or 41 in hexadecimal. Finally, computers store strings as a sequence of bytes. For example, the string "Hello!" composed of six characters can be stored in six bytes: 48 65 6c 6c 6f 21 in hexadecimal.

This is all the computer science concepts needed to understand the rest of this explanation.

B.1 Hash functions

In computer science, a hash function is an algorithm that takes a string of any length (the input) and returns a *fixed-size* sequence of bytes (the output). The output of a hash function is typically called a hash value, hash code, or simply hash, and it is usually represented in hexadecimal. We can view the output of the hash function as a "digest" of the input data.

Figure 7 shows the result of applying different hash functions to an input string. Hash functions are used for various purposes in computing, such as data retrieval, cryptography, and ensuring data integrity [4]. A hash function is usually required to have the following properties to be useful. As we will see later, all of them are critical for its use in Bitcoin.

Deterministic: A hash function must always produce the same output (hash value) for the same input.

Fixed-size output: Regardless of the input data size, the output size of a given hash function is fixed.

Computationally efficient: The hash function should be able to generate a hash value from the input data quickly.

One-way operation: Given a hash value, recovering the original input should be computationally infeasible.

```
Input string:
This is a sample string of text.

MD5 hash function output (output size: 32 hexadecimal digits)
76A1BAE2F3615350A2BAF7B3174C1C28

SHA1 hash function output (output size: 42 hexadecimal digits):
748D38466F757E7AE5C6F7059E99012E6C9BEF9A

SHA256 hash function output (output size: 64 hexadecimal digits):
DB2FB7F0ED98F781B7C86E59F549CCDCB8325B66C4B62C43DA23AE1B02F1FBF1

SHA512 hash function output (output size: 128 hexadecimal digits):
3B76DB49F7DBCB267A05FF9A2A548302F6FA0249DA45A4CB13978A53ED4ED791
53C006E34E440BED8F2CB17AE979069944437D3CD3008AC1D844D34E7AA1349C
```

Figure 7: Application of different hash functions to a given input string.

Avalanche effect: If the input string changes lightly, even a single character, a good hash function should drastically change the hash output.

Collision Resistance: It should be extremely difficult to find two different inputs that produce the same hash output. This property is crucial for avoiding the so-called "hash collisions", especially in cryptographic contexts.

Efficient Distribution: A hash function should distribute hash values uniformly across its possible output range.

Hash functions are frequently used as one-way cryptographic mechanisms. For example, websites usually use hash functions to encrypt passwords. Instead of storing the user's password, which could be stolen, the website applies a hash function to the password and stores the resulting hash, a meaningless sequence of bytes. When the user tries to authenticate again, the website applies the same hash function to the new password and compares it with the stored hash. In this way, no sensitive data is stored in the website's databases.

As we will see shortly, hash functions play a crucial role in how transactions are stored and signed. A related concept is asymmetric encryption, which is briefly introduced in the following section.

B.2 Symmetric and Asymmetric Encryption

As we have seen above, hashing functions are mechanisms that irreversibly encrypt a string. Hashing has its applications, but cannot be used to cipher a message to avoid unwanted reads, because it cannot be reversed.

Symmetric encryption is a classic technique that enables two parties to communicate privately. It can be viewed as a function in which the key is used for encryption and decryption. Suppose that Charlie and Dan want to have a private conversation. In symmetric encryption, both of them should agree to use a common *key* to encrypt and decrypt the message. Note that the same key enables two-way communication: one participant uses it to cipher the message, and the other uses it to decipher it. See Fig. 8(a).

Symmetric encryption, or *private-key encryption*, can be highly secure when appropriately implemented. However, both participants should know the key, and this is precisely the problem: The key should be shared in advance. Suppose that both participants are separated by a long distance. If Charlie generates a new key because the old one was compromised, he should find a way to send it to Dan privately. However, he cannot use the old key to encrypt the message because it is insecure, nor can he use the new key, as Dan does not have it yet. This classical problem is common to all symmetric encryption mechanisms: if a malicious agent intercepts the transmission of the new key, communications are no longer secure.

Asymmetric encryption is a type of cryptography that uses *a pair of keys* to encrypt and decrypt data. One is called the *public key*, and the other is the *private key*. Unlike symmetric encryption, where the same key is used for both encryption and decryption, in asymmetric encryption, one key serves to encrypt the message, and the other one serves to decrypt it. It is essential to note that any

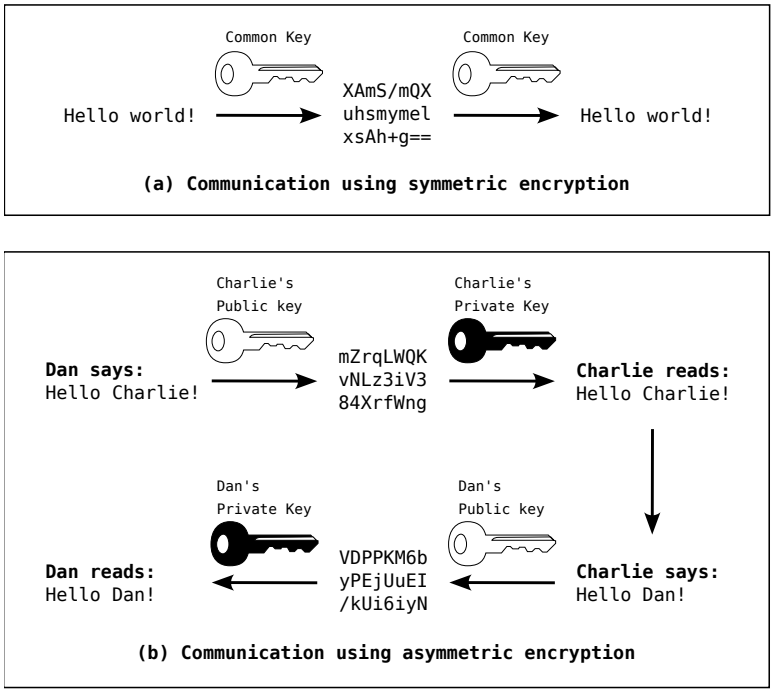

Figure 8: Communication between two partners using (a) symmetric encryption, and (b) asymmetric encryption.

of the keys serves the following purpose: a message encrypted with one of them can be decrypted with the other. However, none can be used in isolation to encrypt and later decrypt the same message. Moreover, the pair of public and private keys is generated in such a way that it is computationally infeasible to derive one of them from the other.

With asymmetric encryption, Charlie and Dan can communicate privately in the following way:

1. Charlie generates a pair of public and private keys. The public key will be used to encrypt the messages directed to Charlie, and Charlie will use the private key to decrypt them.

2. Charlie sends Dan his public key. A public key has no value by itself: when properly used, it simply allows one to translate a string of text into a seemingly random sequence of bytes, just like a hash function does. Therefore, the public key can be sent without any special security measures; this is why it is called a "public key".

3. To privately communicate with Charlie, Dan uses *Charlie's public key* (not Dan's!) to encrypt the message, and send the result to Charlie.

4. Charlie receives the encrypted message and uses his associated secret private key to decrypt it.

It is essential to note that Charlie's keys are only suitable for one-way communication. To allow Charlie to respond, Dan should generate a pair of public and private keys and send Charlie his public key, so Charlie can send Dan an encrypted message while keeping the private key for himself. Figure 8(b) depicts how such a conversation would take place. Note that, unlike symmetric encryption, Charlie's and Dan's public keys can be freely distributed: They are useless without their corresponding private keys.

Asymmetric encryption forms the foundation of modern security systems, including those used for economic transactions such as e-commerce, online banking, secure communication between financial institutions, electronic voting in shareholder meetings, and secure email. As shown in the following section, asymmetric encryption also allows one to produce digital signatures, a key concept in the Bitcoin network, because it demonstrates the ownership of bitcoins.

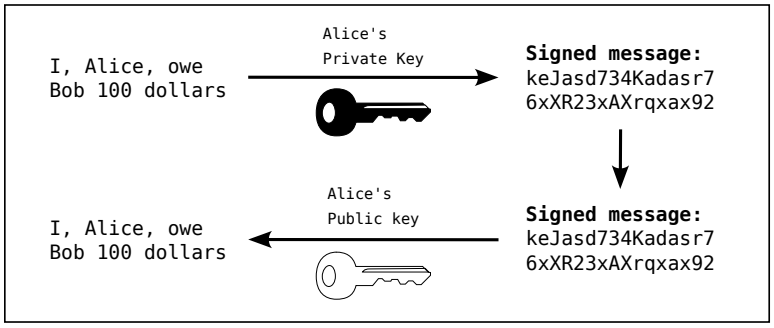

Figure 9: Alice signs a message with her private key. Anyone with Alice's public key can decrypt it.

B.3 Digital signatures

Secure communication is not the only application of asymmetric encryption. A second relevant use of this technique is to produce *digital signatures*. In the real world, a signature is a mechanism that unequivocally states the ownership of the signed object. Asymmetric encryption allows one to produce the equivalent of a signature in the digital world.

Suppose that Alice wants to send a letter stating "I, Alice, owe Bob 100 dollars", and she wants to sign it digitally. To do so, she can apply her private key to the message, and publish (a) the message in plain English, (b) her public key, and (c) the encrypted version, which acts as a signature. As long as the use of private/public keys is reversible, anyone interested in checking that Alice has signed the message can apply Alice's public key to the encrypted message, and he/she will obtain a message that is identical to the copy in plain English, validating the signature. See Fig. 9.

Naturally, anyone could have stolen Alice's keys and signed the message on her behalf. Here, we assume that the person holding the private key is his/her rightful *owner*. As we will see, this observation is crucial to understanding ownership of digital assets, which has profound social implications.

C The mining process

To generate a valid block, miners follow a specific sequence of steps. The procedure is informally described below to illustrate how the blockchain is constructed and, more importantly, why it is considered *immutable*, meaning it is impossible to alter a settled transaction without detection. Many implementation details that are not essential to understanding the process are omitted from this description. The interested reader may consult [1] for additional information.

The steps a miner follows to build a block of pending transactions to be settled can be informally summarized as follows.

1. A certain number of transactions from the mempool is selected. Transactions with the highest fees are more likely to be chosen.
2. For each selected transaction:
   (a) The transaction is re-verified to ensure that signatures and hashes match.
   (b) The current bitcoin balances associated with the transaction's input addresses are retrieved from the blockchain, to ensure that input addresses have not been spent yet.
   (c) The sum of input and output amounts is compared, ensuring the inputs cover the outputs, and that there exists a positive difference, which the miner will take as a fee.
3. The miner starts with a blank block template consisting of a block header and a block body to build the transactions block, and stores all selected transactions as part of the block body.
4. The miner inserts a special transaction, called *Coinbase* transaction, as the first transaction of the block. The Coinbase transaction contains a single output address belonging to the miner. If this block is finally added to the blockchain, this address will receive a certain number of freshly minted bitcoins as a reward, together with the fees of all included transactions.

5. The miner calculates a hash that represents the entire block body. This hash, known as "Merkle Root", is stored in the block header. Note that once the block is built, no changes can be performed to any of its transactions without invalidating this Merkle root.

6. The miner inserts some additional information in the block header, such as the size of the entire block in bytes; the number of transactions included; the order number of this new block in the blockchain (called the *height*), the hash of the header of the last block added to the blockchain (that is, the one at the top), and a timestamp that shows when this block was created.

7. Once the block is built, the miner should solve a challenging puzzle regarding the data stored in this block. This is called "Proof-of-Work".

8. If the miner finds a solution to this puzzle, the miner will announce to the Bitcoin network that a new block "has been found", aiming to say that a new block is ready to be added to the blockchain.

9. Other miners, who were building other blocks and were struggling with their associated Proof-of-Work problem, stop their work and check the new block. Suppose the block is labeled as valid (that is, all the balances, hashes, and the Proof-of-Work are correct). In that case, all the miners remove the newly settled transactions from their set of unconfirmed transactions and restart their work, trying to build a new transaction block on top of the previous one before anybody else.

10. At the same time, Bitcoin nodes that receive the new block of transactions verify it in the same way. If the block is valid, they store it on top of the existing blockchain. If not, the block is discarded.