CHAPTER 2

How Bitcoin Works

Transactions, Blocks, Mining, and the Blockchain

The bitcoin system, unlike traditional banking and payment systems, is based on decentralized trust. Instead of a central trusted authority, in bitcoin, trust is achieved as an emergent property from the interactions of different participants in the bitcoin system. In this chapter, we will examine bitcoin from a high level by tracking a single transaction through the bitcoin system and watch as it becomes "trusted" and accepted by the bitcoin mechanism of distributed consensus and is finally recorded on the blockchain, the distributed ledger of all transactions. Subsequent chapters will delve into the technology behind transactions, the network, and mining.

Bitcoin Overview

In the overview diagram shown in Figure 2-1, we see that the bitcoin system consists of users with wallets containing keys, transactions that are propagated across the network, and miners who produce (through competitive computation) the consensus blockchain, which is the authoritative ledger of all transactions.

Each example in this chapter is based on an actual transaction made on the bitcoin network, simulating the interactions between the users (Joe, Alice, Bob, and Gopesh) by sending funds from one wallet to another. While tracking a transaction through the bitcoin network to the blockchain, we will use a *blockchain explorer* site to visualize each step. A blockchain explorer is a web application that operates as a bitcoin search engine, in that it allows you to search for addresses, transactions, and blocks and see the relationships and flows between them.

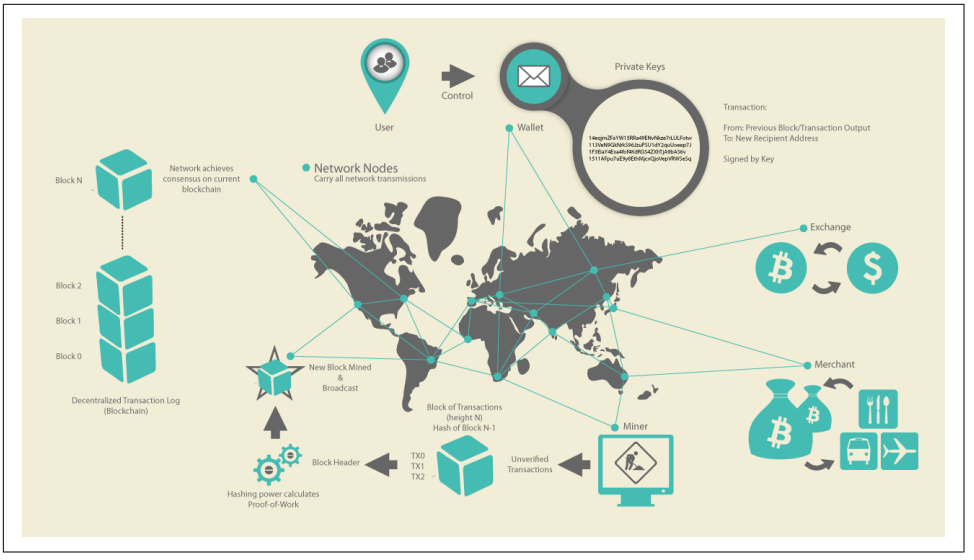

Figure 2-1. Bitcoin overview

Popular blockchain explorers include:

- Bitcoin Block Explorer
- BlockCypher Explorer
- blockchain.info
- BitPay Insight

Each of these has a search function that can take a bitcoin address, transaction hash, block number, or block hash and retrieve corresponding information from the bitcoin network. With each transaction or block example, we will provide a URL so you can look it up yourself and study it in detail.

Buying a Cup of Coffee

Alice, introduced in the previous chapter, is a new user who has just acquired her first bitcoin. In "Getting Your First Bitcoin" on page 10, Alice met with her friend Joe to exchange some cash for bitcoin. The transaction created by Joe funded Alice's wallet with 0.10 BTC. Now Alice will make her first retail transaction, buying a cup of coffee at Bob's coffee shop in Palo Alto, California.

Bob's Cafe recently started accepting bitcoin payments by adding a bitcoin option to its point-of-sale system. The prices at Bob's Cafe are listed in the local currency (US dollars), but at the register, customers have the option of paying in either dollars or bitcoin. Alice places her order for a cup of coffee and Bob enters it into the register, as

he does for all transactions. The point-of-sale system automatically converts the total price from US dollars to bitcoin at the prevailing market rate and displays the price in both currencies:

```
Total:
$1.50 USD
0.015 BTC
```

Bob says, "That's one-dollar-fifty, or fifteen millibits."

Bob's point-of-sale system will also automatically create a special QR code containing a *payment request* (see Figure 2-2).

Unlike a QR code that simply contains a destination bitcoin address, a payment request is a QR-encoded URL that contains a destination address, a payment amount, and a generic description such as "Bob's Cafe." This allows a bitcoin wallet application to prefill the information used to send the payment while showing a human-readable description to the user. You can scan the QR code with a bitcoin wallet application to see what Alice would see.

Figure 2-2. Payment request QR code

Try to scan this with your wallet to see the address and amount but DO NOT SEND MONEY.

```
bitcoin:1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA?
amount=0.015&
label=Bob%27s%20Cafe&
message=Purchase%20at%20Bob%27s%20Cafe

Components of the URL

A bitcoin address: "1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA"
The payment amount: "0.015"
A label for the recipient address: "Bob's Cafe"
A description for the payment: "Purchase at Bob's Cafe"
```

Alice uses her smartphone to scan the barcode on display. Her smartphone shows a payment of `0.0150 BTC` to `Bob's Cafe` and she selects Send to authorize the payment. Within a few seconds (about the same amount of time as a credit card authorization), Bob sees the transaction on the register, completing the transaction.

In the following sections we will examine this transaction in more detail. We'll see how Alice's wallet constructed it, how it was propagated across the network, how it was verified, and finally, how Bob can spend that amount in subsequent transactions.

The bitcoin network can transact in fractional values, e.g., from millibitcoin (1/1000th of a bitcoin) down to 1/100,000,000th of a bitcoin, which is known as a satoshi. Throughout this book we'll use the term "bitcoin" to refer to any quantity of bitcoin currency, from the smallest unit (1 satoshi) to the total number (21,000,000) of all bitcoin that will ever be mined.

You can examine Alice's transaction to Bob's Cafe on the blockchain using a block explorer site (Example 2-1):

Example 2-1. View Alice's transaction on blockexplorer.com

```
https://blockexplorer.com/tx/
0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2
```

Bitcoin Transactions

In simple terms, a transaction tells the network that the owner of some bitcoin value has authorized the transfer of that value to another owner. The new owner can now spend the bitcoin by creating another transaction that authorizes transfer to another owner, and so on, in a chain of ownership.

Transaction Inputs and Outputs

Transactions are like lines in a double-entry bookkeeping ledger. Each transaction contains one or more "inputs," which are like debits against a bitcoin account. On the other side of the transaction, there are one or more "outputs," which are like credits added to a bitcoin account. The inputs and outputs (debits and credits) do not necessarily add up to the same amount. Instead, outputs add up to slightly less than inputs and the difference represents an implied *transaction fee*, which is a small payment collected by the miner who includes the transaction in the ledger. A bitcoin transaction is shown as a bookkeeping ledger entry in Figure 2-3.

The transaction also contains proof of ownership for each amount of bitcoin (inputs) whose value is being spent, in the form of a digital signature from the owner, which

can be independently validated by anyone. In bitcoin terms, "spending" is signing a transaction that transfers value from a previous transaction over to a new owner identified by a bitcoin address.

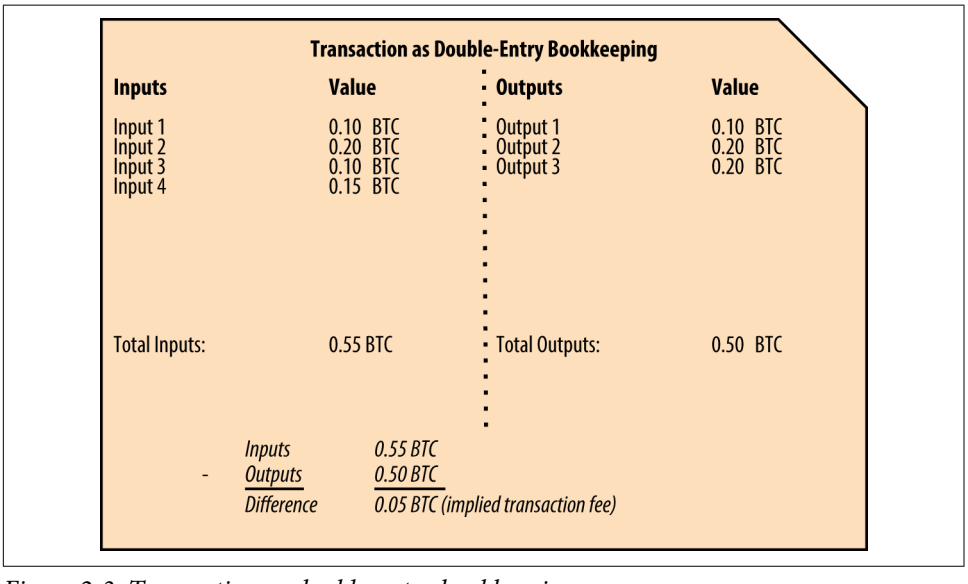

Figure 2-3. Transaction as double-entry bookkeeping

Transaction Chains

Alice's payment to Bob's Cafe uses a previous transaction's output as its input. In the previous chapter, Alice received bitcoin from her friend Joe in return for cash. That transaction created a bitcoin value locked by Alice's key. Her new transaction to Bob's Cafe references the previous transaction as an input and creates new outputs to pay for the cup of coffee and receive change. The transactions form a chain, where the inputs from the latest transaction correspond to outputs from previous transactions. Alice's key provides the signature that unlocks those previous transaction outputs, thereby proving to the bitcoin network that she owns the funds. She attaches the payment for coffee to Bob's address, thereby "encumbering" that output with the requirement that Bob produces a signature in order to spend that amount. This represents a transfer of value between Alice and Bob. This chain of transactions, from Joe to Alice to Bob, is illustrated in Figure 2-4.

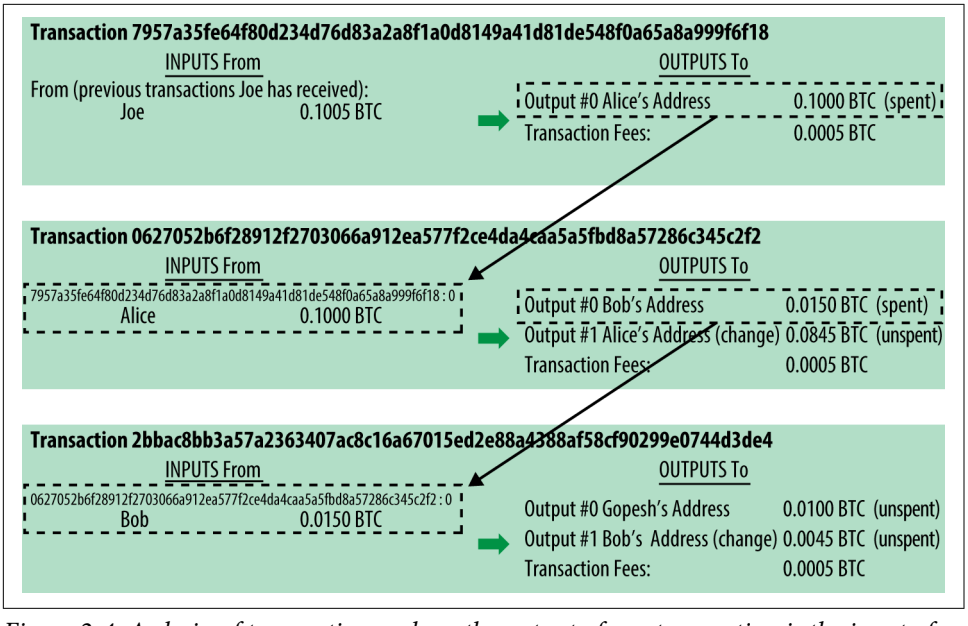

Figure 2-4. A chain of transactions, where the output of one transaction is the input of the next transaction

Making Change

Many bitcoin transactions will include outputs that reference both an address of the new owner and an address of the current owner, called the *change* address. This is because transaction inputs, like currency notes, cannot be divided. If you purchase a $5 US dollar item in a store but use a $20 US dollar bill to pay for the item, you expect to receive $15 US dollars in change. The same concept applies with bitcoin transaction inputs. If you purchased an item that costs 5 bitcoin but only had a 20 bitcoin input to use, you would send one output of 5 bitcoin to the store owner and one output of 15 bitcoin back to yourself as change (less any applicable transaction fee). Importantly, the change address does not have to be the same address as that of the input and for privacy reasons is often a new address from the owner's wallet.

Different wallets may use different strategies when aggregating inputs to make a payment requested by the user. They might aggregate many small inputs, or use one that is equal to or larger than the desired payment. Unless the wallet can aggregate inputs in such a way to exactly match the desired payment plus transaction fees, the wallet will need to generate some change. This is very similar to how people handle cash. If you always use the largest bill in your pocket, you will end up with a pocket full of loose change. If you only use the loose change, you'll always have only big bills. People subconsciously find a balance between these two extremes, and bitcoin wallet developers strive to program this balance.

In summary, *transactions* move value from *transaction inputs* to *transaction outputs*. An input is a reference to a previous transaction's output, showing where the value is coming from. A transaction output directs a specific value to a new owner's bitcoin address and can include a change output back to the original owner. Outputs from one transaction can be used as inputs in a new transaction, thus creating a chain of ownership as the value is moved from owner to owner (see Figure 2-4).

Common Transaction Forms

The most common form of transaction is a simple payment from one address to another, which often includes some "change" returned to the original owner. This type of transaction has one input and two outputs and is shown in Figure 2-5.

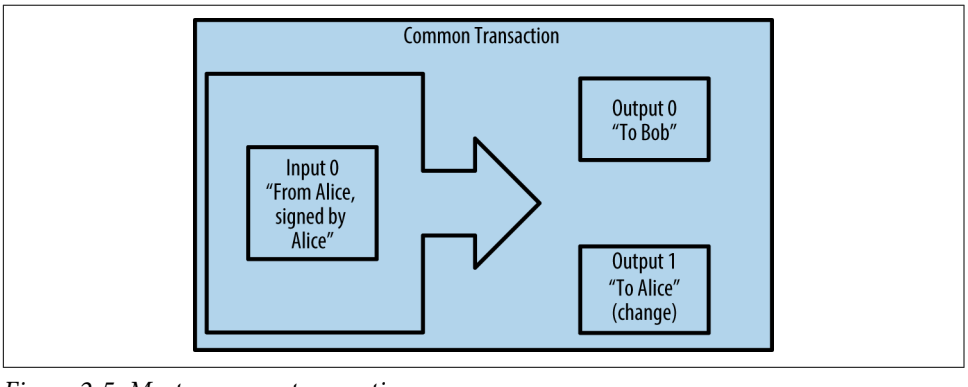

Figure 2-5. Most common transaction

Another common form of transaction is one that aggregates several inputs into a single output (see Figure 2-6). This represents the real-world equivalent of exchanging a pile of coins and currency notes for a single larger note. Transactions like these are sometimes generated by wallet applications to clean up lots of smaller amounts that were received as change for payments.

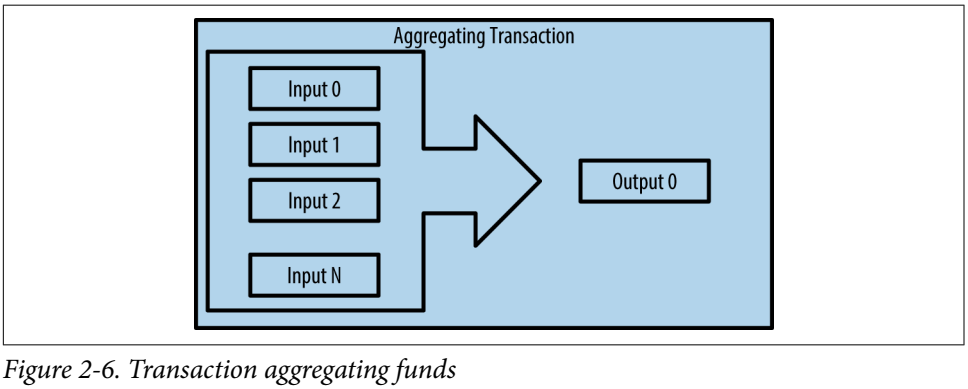

Figure 2-6. Transaction aggregating funds

Finally, another transaction form that is seen often on the bitcoin ledger is a transaction that distributes one input to multiple outputs representing multiple recipients (see Figure 2-7). This type of transaction is sometimes used by commercial entities to distribute funds, such as when processing payroll payments to multiple employees.

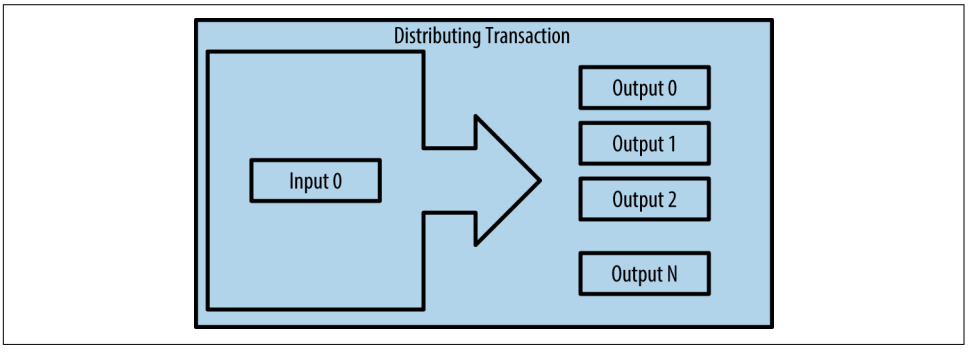

Figure 2-7. Transaction distributing funds

Constructing a Transaction

Alice's wallet application contains all the logic for selecting appropriate inputs and outputs to build a transaction to Alice's specification. Alice only needs to specify a destination and an amount, and the rest happens in the wallet application without her seeing the details. Importantly, a wallet application can construct transactions even if it is completely offline. Like writing a check at home and later sending it to the bank in an envelope, the transaction does not need to be constructed and signed while connected to the bitcoin network.

Getting the Right Inputs

Alice's wallet application will first have to find inputs that can pay for the amount she wants to send to Bob. Most wallets keep track of all the available outputs belonging to addresses in the wallet. Therefore, Alice's wallet would contain a copy of the transaction output from Joe's transaction, which was created in exchange for cash (see "Getting Your First Bitcoin" on page 10). A bitcoin wallet application that runs as a full-node client actually contains a copy of every unspent output from every transaction in the blockchain. This allows a wallet to construct transaction inputs as well as quickly verify incoming transactions as having correct inputs. However, because a full-node client takes up a lot of disk space, most user wallets run "lightweight" clients that track only the user's own unspent outputs.

If the wallet application does not maintain a copy of unspent transaction outputs, it can query the bitcoin network to retrieve this information using a variety of APIs available by different providers or by asking a full-node using an application pro-

gramming interface (API) call. Example 2-2 shows a API request, constructed as an HTTP GET command to a specific URL. This URL will return all the unspent transaction outputs for an address, giving any application the information it needs to construct transaction inputs for spending. We use the simple command-line HTTP client *cURL* to retrieve the response.

Example 2-2. Look up all the unspent outputs for Alice's bitcoin address

```
$ curl https://blockchain.info/unspent?active=1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK

    {

            "unspent_outputs":[

                    {
                            "tx_hash":"186f9f998a5...2836dd734d2804fe65fa35779",
                            "tx_index":104810202,
                            "tx_output_n": 0,

    "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
                            "value": 10000000,
                            "value_hex": "00989680",
                            "confirmations":0
                    }

            ]
    }
```

The response in Example 2-2 shows one unspent output (one that has not been redeemed yet) under the ownership of Alice's address `1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK`. The response includes the reference to the transaction in which this unspent output is contained (the payment from Joe) and its value in satoshis, at 10 million, equivalent to 0.10 bitcoin. With this information, Alice's wallet application can construct a transaction to transfer that value to new owner addresses.

View the transaction from Joe to Alice.

As you can see, Alice's wallet contains enough bitcoin in a single unspent output to pay for the cup of coffee. Had this not been the case, Alice's wallet application might have to "rummage" through a pile of smaller unspent outputs, like picking coins from a purse until it could find enough to pay for the coffee. In both cases, there might be a need to get some change back, which we will see in the next section, as the wallet application creates the transaction outputs (payments).

Creating the Outputs

A transaction output is created in the form of a script that creates an encumbrance on the value and can only be redeemed by the introduction of a solution to the script. In simpler terms, Alice's transaction output will contain a script that says something like, "This output is payable to whoever can present a signature from the key corresponding to Bob's public address." Because only Bob has the wallet with the keys corresponding to that address, only Bob's wallet can present such a signature to redeem this output. Alice will therefore "encumber" the output value with a demand for a signature from Bob.

This transaction will also include a second output, because Alice's funds are in the form of a 0.10 BTC output, too much money for the 0.015 BTC cup of coffee. Alice will need 0.085 BTC in change. Alice's change payment is created by Alice's wallet as an output in the very same transaction as the payment to Bob. Essentially, Alice's wallet breaks her funds into two payments: one to Bob and one back to herself. She can then use (spend) the change output in a subsequent transaction.

Finally, for the transaction to be processed by the network in a timely fashion, Alice's wallet application will add a small fee. This is not explicit in the transaction; it is implied by the difference between inputs and outputs. If instead of taking 0.085 in change, Alice creates only 0.0845 as the second output, there will be 0.0005 BTC (half a millibitcoin) left over. The input's 0.10 BTC is not fully spent with the two outputs, because they will add up to less than 0.10. The resulting difference is the *transaction fee* that is collected by the miner as a fee for validating and including the transaction in a block to be recorded on the blockchain.

The resulting transaction can be seen using a blockchain explorer web application, as shown in Figure 2-8.

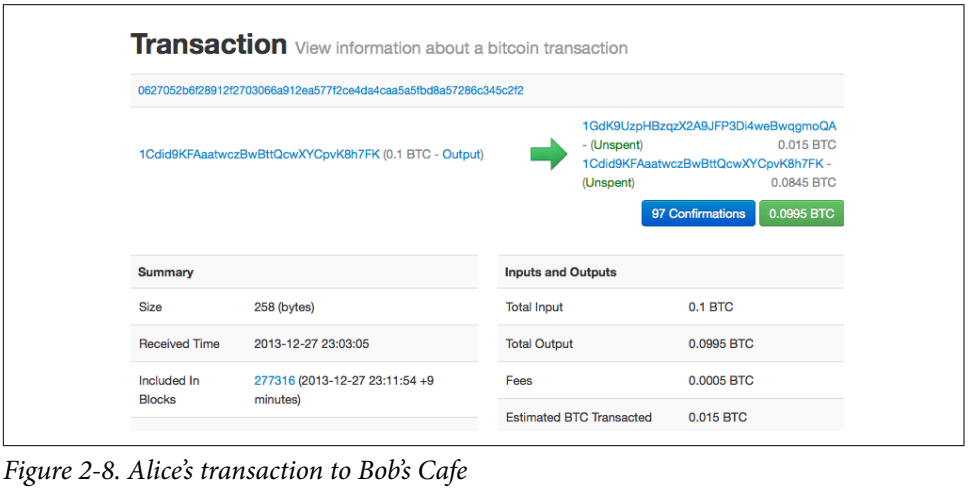

Figure 2-8. Alice's transaction to Bob's Cafe

View the transaction from Alice to Bob's Cafe.

Adding the Transaction to the Ledger

The transaction created by Alice's wallet application is 258 bytes long and contains everything necessary to confirm ownership of the funds and assign new owners. Now, the transaction must be transmitted to the bitcoin network where it will become part of the blockchain. In the next section we will see how a transaction becomes part of a new block and how the block is "mined." Finally, we will see how the new block, once added to the blockchain, is increasingly trusted by the network as more blocks are added.

Transmitting the transaction

Because the transaction contains all the information necessary to process, it does not matter how or where it is transmitted to the bitcoin network. The bitcoin network is a peer-to-peer network, with each bitcoin client participating by connecting to several other bitcoin clients. The purpose of the bitcoin network is to propagate transactions and blocks to all participants.

How it propagates

Any system, such as a server, desktop application, or wallet, that participates in the bitcoin network by "speaking" the bitcoin protocol is called a *bitcoin node*. Alice's wallet application can send the new transaction to any bitcoin node it is connected to over any type of connection: wired, WiFi, mobile, etc. Her bitcoin wallet does not have to be connected to Bob's bitcoin wallet directly and she does not have to use the internet connection offered by the cafe, though both those options are possible, too. Any bitcoin node that receives a valid transaction it has not seen before will immediately forward it to all other nodes to which it is connected, a propagation technique known as *flooding*. Thus, the transaction rapidly propagates out across the peer-to-peer network, reaching a large percentage of the nodes within a few seconds.

Bob's view

If Bob's bitcoin wallet application is directly connected to Alice's wallet application, Bob's wallet application might be the first node to receive the transaction. However, even if Alice's wallet sends the transaction through other nodes, it will reach Bob's wallet within a few seconds. Bob's wallet will immediately identify Alice's transaction as an incoming payment because it contains outputs redeemable by Bob's keys. Bob's wallet application can also independently verify that the transaction is well formed,

uses previously unspent inputs, and contains sufficient transaction fees to be included in the next block. At this point Bob can assume, with little risk, that the transaction will shortly be included in a block and confirmed.

A common misconception about bitcoin transactions is that they must be "confirmed" by waiting 10 minutes for a new block, or up to 60 minutes for a full six confirmations. Although confirmations ensure the transaction has been accepted by the whole network, such a delay is unnecessary for small-value items such as a cup of coffee. A merchant may accept a valid small-value transaction with no confirmations, with no more risk than a credit card payment made without an ID or a signature, as merchants routinely accept today.

Bitcoin Mining

Alice's transaction is now propagated on the bitcoin network. It does not become part of the *blockchain* until it is verified and included in a block by a process called *mining*. See Chapter 10 for a detailed explanation.

The bitcoin system of trust is based on computation. Transactions are bundled into *blocks*, which require an enormous amount of computation to prove, but only a small amount of computation to verify as proven. The mining process serves two purposes in bitcoin:

- Mining nodes validate all transactions by reference to bitcoin's *consensus rules*. Therefore, mining provides security for bitcoin transactions by rejecting invalid or malformed transactions.
- Mining creates new bitcoin in each block, almost like a central bank printing new money. The amount of bitcoin created per block is limited and diminishes with time, following a fixed issuance schedule.

Mining achieves a fine balance between cost and reward. Mining uses electricity to solve a mathematical problem. A successful miner will collect a *reward* in the form of new bitcoin and transaction fees. However, the reward will only be collected if the miner has correctly validated all the transactions, to the satisfaction of the rules of *consensus*. This delicate balance provides security for bitcoin without a central authority.

A good way to describe mining is like a giant competitive game of sudoku that resets every time someone finds a solution and whose difficulty automatically adjusts so that it takes approximately 10 minutes to find a solution. Imagine a giant sudoku puzzle, several thousand rows and columns in size. If I show you a completed puzzle you can verify it quite quickly. However, if the puzzle has a few squares filled and the rest

are empty, it takes a lot of work to solve! The difficulty of the sudoku can be adjusted by changing its size (more or fewer rows and columns), but it can still be verified quite easily even if it is very large. The "puzzle" used in bitcoin is based on a cryptographic hash and exhibits similar characteristics: it is asymmetrically hard to solve but easy to verify, and its difficulty can be adjusted.

In "Bitcoin Uses, Users, and Their Stories" on page 5, we introduced Jing, an entrepreneur in Shanghai. Jing runs a *mining farm*, which is a business that runs thousands of specialized mining computers, competing for the reward. Every 10 minutes or so, Jing's mining computers compete against thousands of similar systems in a global race to find a solution to a block of transactions. Finding such a solution, the so-called *Proof-of-Work* (PoW), requires quadrillions of hashing operations per second across the entire bitcoin network. The algorithm for Proof-of-Work involves repeatedly hashing the header of the block and a random number with the SHA256 cryptographic algorithm until a solution matching a predetermined pattern emerges. The first miner to find such a solution wins the round of competition and publishes that block into the blockchain.

Jing started mining in 2010 using a very fast desktop computer to find a suitable Proof-of-Work for new blocks. As more miners started joining the bitcoin network, the difficulty of the problem increased rapidly. Soon, Jing and other miners upgraded to more specialized hardware, such as high-end dedicated graphical processing units (GPUs) cards such as those used in gaming desktops or consoles. At the time of this writing, the difficulty is so high that it is profitable only to mine with application-specific integrated circuits (ASIC), essentially hundreds of mining algorithms printed in hardware, running in parallel on a single silicon chip. Jing's company also participates in a *mining pool*, which much like a lottery pool allows several participants to share their efforts and rewards. Jing's company now runs a warehouse containing thousands of ASIC miners to mine for bitcoin 24 hours a day. The company pays its electricity costs by selling the bitcoin it is able to generate from mining, creating some income from the profits.

Mining Transactions in Blocks

New transactions are constantly flowing into the network from user wallets and other applications. As these are seen by the bitcoin network nodes, they get added to a temporary pool of unverified transactions maintained by each node. As miners construct a new block, they add unverified transactions from this pool to the new block and then attempt to prove the validity of that new block, with the mining algorithm (Proof-of-Work). The process of mining is explained in detail in Chapter 10.

Transactions are added to the new block, prioritized by the highest-fee transactions first and a few other criteria. Each miner starts the process of mining a new block of transactions as soon as he receives the previous block from the network, knowing he

has lost that previous round of competition. He immediately creates a new block, fills it with transactions and the fingerprint of the previous block, and starts calculating the Proof-of-Work for the new block. Each miner includes a special transaction in his block, one that pays his own bitcoin address the block reward (currently 12.5 newly created bitcoin) plus the sum of transaction fees from all the transactions included in the block. If he finds a solution that makes that block valid, he "wins" this reward because his successful block is added to the global blockchain and the reward transaction he included becomes spendable. Jing, who participates in a mining pool, has set up his software to create new blocks that assign the reward to a pool address. From there, a share of the reward is distributed to Jing and other miners in proportion to the amount of work they contributed in the last round.

Alice's transaction was picked up by the network and included in the pool of unverified transactions. Once validated by the mining software it was included in a new block, called a *candidate block*, generated by Jing's mining pool. All the miners participating in that mining pool immediately start computing Proof-of-Work for the candidate block. Approximately five minutes after the transaction was first transmitted by Alice's wallet, one of Jing's ASIC miners found a solution for the candidate block and announced it to the network. Once other miners validated the winning block they started the race to generate the next block.

Jing's winning block became part of the blockchain as block #277316, containing 420 transactions, including Alice's transaction. The block containing Alice's transaction is counted as one "confirmation" of that transaction.

You can see the block that includes Alice's transaction.

Approximately 19 minutes later, a new block, #277317, is mined by another miner. Because this new block is built on top of block #277316 that contained Alice's transaction, it added even more computation to the blockchain, thereby strengthening the trust in those transactions. Each block mined on top of the one containing the transaction counts as an additional confirmation for Alice's transaction. As the blocks pile on top of each other, it becomes exponentially harder to reverse the transaction, thereby making it more and more trusted by the network.

In the diagram in Figure 2-9, we can see block #277316, which contains Alice's transaction. Below it are 277,316 blocks (including block #0), linked to each other in a chain of blocks (blockchain) all the way back to block #0, known as the *genesis block*. Over time, as the "height" in blocks increases, so does the computation difficulty for each block and the chain as a whole. The blocks mined after the one that contains

Alice's transaction act as further assurance, as they pile on more computation in a longer and longer chain. By convention, any block with more than six confirmations is considered irrevocable, because it would require an immense amount of computation to invalidate and recalculate six blocks. We will examine the process of mining and the way it builds trust in more detail in Chapter 10.

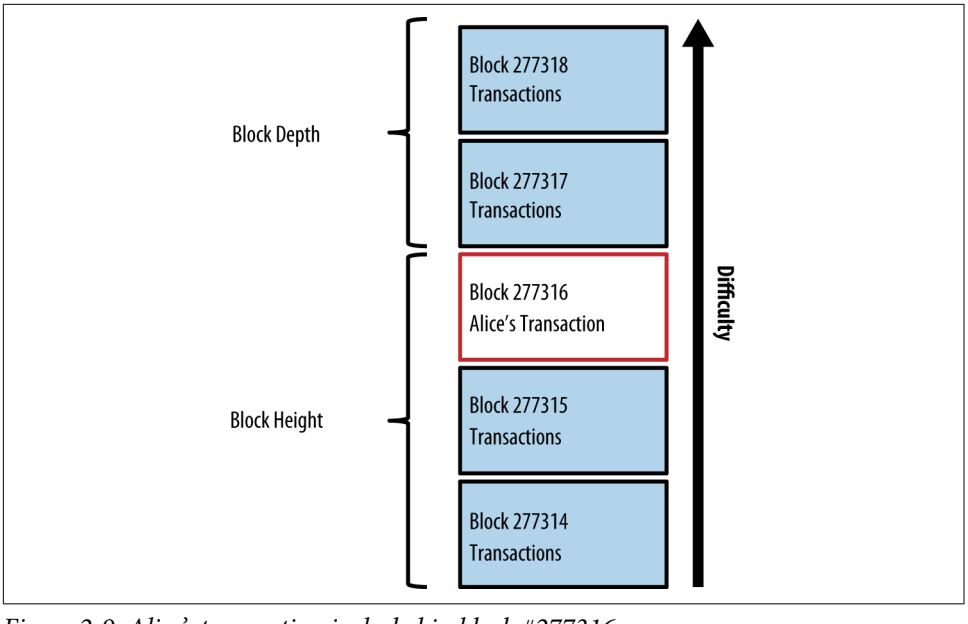

Figure 2-9. Alice's transaction included in block #277316

Spending the Transaction

Now that Alice's transaction has been embedded in the blockchain as part of a block, it is part of the distributed ledger of bitcoin and visible to all bitcoin applications. Each bitcoin client can independently verify the transaction as valid and spendable. Full-node clients can track the source of the funds from the moment the bitcoin were first generated in a block, incrementally from transaction to transaction, until they reach Bob's address. Lightweight clients can do what is called a simplified payment verification (see "Simplified Payment Verification (SPV) Nodes" on page 183) by confirming that the transaction is in the blockchain and has several blocks mined after it, thus providing assurance that the miners accepted it as valid.

Bob can now spend the output from this and other transactions. For example, Bob can pay a contractor or supplier by transferring value from Alice's coffee cup payment to these new owners. Most likely, Bob's bitcoin software will aggregate many small payments into a larger payment, perhaps concentrating all the day's bitcoin revenue into a single transaction. This would aggregate the various payments into a single

output (and a single address). For a diagram of an aggregating transaction, see Figure 2-6.

As Bob spends the payments received from Alice and other customers, he extends the chain of transactions. Let's assume that Bob pays his web designer Gopesh in Bangalore for a new website page. Now the chain of transactions will look like Figure 2-10.

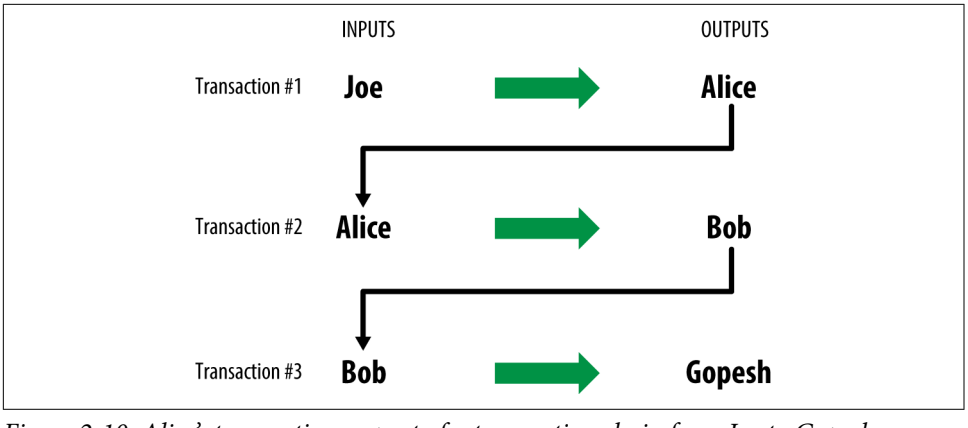

Figure 2-10. Alice's transaction as part of a transaction chain from Joe to Gopesh

In this chapter, we saw how transactions build a chain that moves value from owner to owner. We also tracked Alice's transaction, from the moment it was created in her wallet, through the bitcoin network and to the miners who recorded it on the blockchain. In the rest of this book we will examine the specific technologies behind wallets, addresses, signatures, transactions, the network, and finally mining.