Chapter 16 Digital Cash

Suppose Congressman Bill Passer is receiving large donations from his friend Phil Pockets. For obvious reasons, he would like to hide this fact, pretending instead that the money comes mostly from people such as Vera Goode. Or perhaps Phil does not want Bill to know he’s the source of the money. If Phil pays by check, well-placed sources in the bank can expose him. Similarly, Congressman Passer cannot receive payments via credit card. The only anonymous payment scheme seems to be cash.

But now suppose Passer has remained in office for many terms and we are nearing the end of the twenty-first century. All commerce is carried out electronically. Is it possible to have electronic cash? Several problems arise. For example, near the beginning of the twenty-first century, photocopying money was possible, though a careful recipient could discern differences between the copy and the original. Copies of electronic information, however, are indistinguishable from the original. Therefore, someone who has a valid electronic coin could make several copies. Some method is needed to prevent such double spending. One idea would be for a central bank to have records of every coin and who has each one. But if coins are recorded as they are spent, anonymity is compromised. Occasionally, communications with a central bank could fail temporarily, so it is also desirable for the person receiving the coin to be able to verify the coin as legitimate without contacting the bank during each transaction.

T. Okamoto and K. Ohta [Okamoto-Ohta] list six properties a digital cash system should have:

1. The cash can be sent securely through computer networks.
2. The cash cannot be copied and reused.
3. The spender of the cash can remain anonymous. If the coin is spent legitimately, neither the recipient nor the bank can identify the spender.
4. The transaction can be done *off-line*, meaning no communication with the central bank is needed during the transaction.
5. The cash can be transferred to others.
6. A piece of cash can be divided into smaller amounts.

Okamoto and Ohta gave a system that satisfies all these requirements. Several systems satisfying some of them have been devised by David Chaum and others. In Section 16.2, we describe a system due to S. Brands [Brands] that satisfies 1 through 4. We include it to show the complicated manipulations that are used to achieve these goals. But an underlying basic problem is that it and the Okamoto-Ohta system require a central bank to set up the system. This limits their use.

In Section 16.3, we give an introduction to Bitcoin, a well-known digital currency. By ingeniously relaxing the requirements of Okamoto-Ohta, it discarded the need for a central bank and therefore became much more widely used than its predecessors.

16.1 Setting the Stage for Digital Economies

People have always had a need to engage in trade in order to acquire items that they do not have. The history of commerce has evolved from the ancient model of barter and exchange to notions of promises and early forms of credit ("If you give me bread today, then I'll give you milk in the future"), to the use of currencies and cash, to the merging of currencies and credit in the form of credit cards. All of these have changed over time, and recent shifts toward conducting transactions electronically have forced these technologies to evolve significantly.

Perhaps the most dominant way in which electronic transactions take place is with the use of credit cards. We are all familiar with making purchases at stores using conventional credit cards: our items are scanned at a register, a total bill is presented, and we pay by swiping our card (or inserting a card with a chip) at a credit card scanner. Using credit cards online is not too different. When you want to buy something online from eVendor, you enter your credit card number along with some additional information (e.g., the CVC, expiration date, address). Whether you use a computer, a smartphone, or any other type of device, there is a secure communication protocol working behind the scenes that sends this information to eVendor, and secure protocols support eVendor by contacting the proper banks and credit card agencies to authorize and complete your transaction.

While using credit cards is extremely easy, there are problems with their use in a world that has been increasingly digital. The early 21st century has seen many examples of companies being hacked and credit card information being stolen. A further problem with the use of credit cards is that companies have the ability to track customer purchases and preferences and, as a result, issues of consumer privacy are becoming more prevalent.

There are alternatives to the use of credit cards. One example is the introduction of an additional layer between the consumer and the vendor. Companies such as PayPal provide such a service. They interact with eVendor, providing a guarantee that eVendor will get paid, while also ensuring that eVendor does not learn who you are. Of course, such a solution begs the question of how much trust one should place in these intermediate companies.

A different alternative comes from looking at society’s use of hard, tangible currencies. Coins and cash have some very nice properties when one considers their use from a security and privacy perspective. First, since they are physical objects representing real value, they provide an immediate protection against any defaulting or credit risk. If Alice wants an object that costs five dollars and she has five dollars, then there is no need for a credit card or an I-Owe-You. The vendor can complete the transaction knowing that he or she has definitely acquired five dollars. Second, cash and coins are not tied to the individual using them, so they provide strong anonymity protection. The vendor doesn’t care about Alice or her identity; it only cares about getting the money associated with the transaction. Likewise, there are no banks directly involved in a transaction. Currency is printed by a central government, and this currency is somehow backed by the government in one way or another.

Cash and coins also have the nice property that they are actually exchanged in a transaction. By this, we mean that when Alice spends her five dollars, she has handed over the money and she is no longer in possession of this money. This means that she can’t spend the same money over and over. Lastly, because of the physical nature of cash and coins, there is no need to communicate with servers and banks to complete a transaction, which allows for transactions to be completed off-line.

In this chapter, we will also discuss the design of digital currencies, starting from one of the early models for digital coins and then exploring the more recent forms of cryptocurrencies by examining the basic cryptographic constructions used in Bitcoin. As we shall see, many of the properties that we take for granted with cash and coins have been particularly difficult to achieve in the digital world.

16.2 A Digital Cash System

This section is not needed for the remaining sections of the chapter. It is included because it has some interesting ideas and it shows how hard it is to achieve the desired requirements of a digital currency.

We describe a system due to S. Brands [Brands]. The reader will surely notice that it is much more complicated than the centuries-old system of actual coins. This is because, as we mentioned previously, electronic objects can be reproduced at essentially no cost, in contrast to physical cash, which has usually been rather difficult to counterfeit. Therefore, steps are needed to catch electronic cash counterfeiters. But this means that something like a user's signature needs to be attached to an electronic coin. How, then, can anonymity be preserved? The solution uses "restricted blind signatures." This process contributes much of the complexity to the scheme.

16.2.1 Participants

Participants are the Bank, the Spender, and the Merchant.

16.2.2 Initialization

Initialization is done once and for all by some central authority. Choose a large prime p such that $q = (p-1)/2$ is also prime (see Exercise 15 in Chapter 13). Let g be the square of a primitive root mod p. This implies that

$g^{k_1} \equiv g^{k_2} \pmod{p} \iff k_1 \equiv k_2 \pmod{q}$. Two secret random exponents are chosen, and g_1 and g_2 are defined to be g raised to these exponents mod p. These exponents are then discarded (storing them serves no useful purpose, and if a hacker discovers them, then the system is compromised). The numbers

$$g, \quad g_1, \quad g_2$$

are made public. Also, two public hash functions are chosen. The first, H, takes a 5-tuple of integers as input and outputs an integer mod q. The second, H_0, takes a 4-tuple of integers as input and outputs an integer mod q .

16.2.3 The Bank

The bank chooses its secret identity number x and computes

$$h \equiv g^x \pmod{p}.$$

The number h is made public and identifies the bank.

16.2.4 The Spender

The Spender chooses a secret identity number u and computes the account number

$$I \equiv g_1^u \pmod{p}.$$

The number I is sent to the Bank, which stores I along with information identifying the Spender (e.g., name, address). However, the Spender does not send u to the bank. The Bank sends

$$z' \equiv (I g_2)^x \pmod{p}$$

to the Spender.

16.2.5 The Merchant

The Merchant chooses an identification number M and registers it with the bank.

16.2.6 Creating a Coin

The Spender contacts the bank, asking for a coin. The bank requires proof of identity, just as when someone is withdrawing classical cash from an account. All coins in the present scheme have the same value. A coin will be represented by a 6-tuple of numbers

$$(A, B, z, a, b, r).$$

This may seem overly complicated, but we'll see that most of this effort is needed to preserve anonymity and at the same time prevent double spending.

Here is how the numbers are constructed.

1. The Bank chooses a random number w (a different number for each coin), computes

   $$g_w \equiv g^w \text{ and } \beta \equiv (Ig_2)^w \pmod{p},$$

   and sends g_w and β to the Spender.

2. The Spender chooses a secret random 5-tuple of integers

   $$(s, x_1, x_2, \alpha_1, \alpha_2).$$

3. The Spender computes

   $$A \equiv (Ig_2)^s, \quad B \equiv g_1^{x_1} g_2^{x_2}, \quad z \equiv z'^s,$$
   $$a \equiv g_w^{\alpha_1} g^{\alpha_2}, \quad b \equiv \beta^{s\alpha_1} A^{\alpha_2} \pmod{p}.$$

   Coins with $A = 1$ are not allowed. This can happen in only two ways. One is when $s \equiv 0 \pmod{q}$, so we require $s \not\equiv 0$. The other is when $Ig_2 \equiv 1 \pmod{p}$, which means the Spender has solved a discrete logarithm problem by a lucky choice of u. The prime p should be chosen so large that this has essentially no chance of happening.

4. The Spender computes

   $$c \equiv \alpha_1^{-1} H(A, B, z, a, b) \pmod{q}$$

and sends c to the Bank. Here H is the public hash function mentioned earlier.

5. The Bank computes $c_1 \equiv cx + w \pmod{q}$ and sends c_1 to the Spender.

6. The Spender computes

$$r \equiv \alpha_1 c_1 + \alpha_2 \pmod{q}.$$

   The coin (A, B, z, a, b, r) is now complete. The amount of the coin is deducted from the Spender's bank account.

The procedure, which is quite fast, is repeated each time a Spender wants a coin. A new random number w should be chosen by the Bank for each transaction. Similarly, each spender should choose a new 5-tuple $(s, x_1, x_2, \alpha_1, \alpha_2)$ for each coin.

16.2.7 Spending the Coin

The Spender gives the coin (A, B, z, a, b, r) to the Merchant. The following procedure is then performed:

1. The Merchant checks whether

$$g^r \equiv a\, h^{H(A,\, B,\, z,\, a,\, b)} \quad A^r \equiv z^{H(A,\, B,\, z,\, a,\, b)} b \pmod{p}.$$

   If this is the case, the Merchant knows that the coin is valid. However, more steps are required to prevent double spending.

2. The Merchant computes

$$d = H_0(A, B, M, t),$$

   where H_0 is the hash function chosen in the initialization phase and t is a number representing the date and time of the transaction. The number t is included so that different transactions will have different values of d. The Merchant sends d to the Spender.

3. The Spender computes

$$r_1 \equiv dus + x_1, \quad r_2 \equiv ds + x_2 \pmod{q},$$

   where u is the Spender's secret number, and s, x_1, x_2 are part of the secret random 5-tuple chosen earlier. The Spender sends r_1 and r_2 to the Merchant.

4. The Merchant checks whether

$$g_1^{r_1} g_2^{r_2} \equiv A^d B \pmod{p}.$$

If this congruence holds, the Merchant accepts the coin. Otherwise, the Merchant rejects it.

16.2.8 The Merchant Deposits the Coin in the Bank

A few days after receiving the coin, the Merchant wants to deposit it in the Bank. The Merchant submits the coin (A, B, z, a, b, r) plus the triple (r_1, r_2, d). The Bank performs the following:

1. The Bank checks that the coin (A, B, z, a, b, r) has not been previously deposited. If it hasn't been, then the next step is performed. If it has been previously deposited, the Bank skips to the Fraud Control procedures discussed in the next subsection.
2. The Bank checks that

   $$g^r \equiv a\, h^{H(A,\, B,\, z,\, a,\, b)} \quad A^r \equiv z^{H(A,\, B,\, z,\, a,\, b)} b, \text{ and } g_1^{r_1} g_2^{r_2} \equiv A^d B \pmod{p}.$$

   If so, the coin is valid and the Merchant's account is credited.

16.2.9 Fraud Control

There are several possible ways for someone to try to cheat. Here is how they are dealt with.

1. The Spender spends the coin twice, once with the Merchant, and once with someone we'll call the Vendor. The Merchant submits the coin along with the triple (r_1, r_2, d). The Vendor submits the coin along with the triple (r_1', r_2', d'). An easy calculation shows that

   $$r_1 - r_1' \equiv us(d - d'), \quad r_2 - r_2' \equiv s(d - d') \pmod{q}.$$

   Dividing yields $u \equiv (r_1 - r_1')(r_2 - r_2')^{-1} \pmod{q}$. The Bank computes $I \equiv g_1^u \pmod{p}$ and identifies the Spender. Since the Bank cannot discover u otherwise, it has proof (at least beyond a reasonable doubt) that double spending has occurred. The Spender is then sent to jail (if the jury believes that the discrete logarithm problem is hard).

2. The Merchant tries submitting the coin twice, once with the legitimate triple (r_1, r_2, d) and once with a forged triple $(r_1^{'}, r_2^{'}, d')$. This is essentially impossible for the Merchant to do, since it is very difficult for the Merchant to produce numbers such that

$$g_1^{r_1^{'}} g_2^{r_2^{'}} \equiv A^{d'} B \pmod{p}.$$

3. Someone tries to make an unauthorized coin. This requires finding numbers such that $g^r \equiv a\, h^{H(A,\, B,\, z,\, a,\, b)}$ and $A^r \equiv z^{H(A,\, B,\, z,\, a,\, b)} b$. This is probably hard to do. For example, starting with $A,\, B,\, z,\, a,\, b$, then trying to find r, requires solving a discrete logarithm problem just to make the first equation work. Note that the Spender is foiled in attempts to produce a second coin using a new 5-tuple since the values of x is known only to the Bank. Therefore, finding the correct value of r is very difficult.

4. Eve L. Dewar, an evil merchant, receives a coin from the Spender and deposits it in the bank, but also tries to spend the coin with the Merchant. Eve gives the coin to the Merchant, who computes d', which very likely is not equal to d. Eve does not know $u,\, x_1,\, x_2,\, s$, but she must choose $r_1^{'}$ and $r_2^{'}$ such that $g_1^{r_1^{'}} g_2^{r_2^{'}} \equiv A^{d'} B \pmod{p}$. This again is a type of discrete logarithm problem. Why can't Eve simply use the $r_1,\, r_2$ that she already knows? Since $d' \neq d$, the Merchant would find that $g_1^{r_1} g_2^{r_2} \not\equiv A^{d'} B$.

5. Someone working in the Bank tries to forge a coin. This person has essentially the same information as Eve, plus the identification number I. It is possible to make a coin that satisfies $g^r \equiv a\, h^{H(A,\, B,\, z,\, a,\, b)}$. However, since the Spender has kept u secret, the person in the bank will not be able to produce a suitable r_1. Of course, if $s = 0$ were allowed, this would be possible; this is one reason $A = 1$ is not allowed.

6. Someone steals the coin from the Spender and tries to spend it. The first verification equation is still satisfied, but the thief does not know u and therefore will not be able to produce $r_1,\, r_2$ such that $g_1^{r_1} g_2^{r_2} \equiv A^{d'} B$.

7. Eve L. Dewar, the evil merchant, steals the coin and $(r_1,\, r_2,\, d)$ from the Merchant before they are submitted to the Bank. Unless the bank requires merchants to keep records of the time and date of each transaction, and therefore be able to reproduce the inputs that produced d, Eve's theft will be successful. This, of course, is a flaw of ordinary cash, too.

16.2.10 Anonymity

During the entire transaction with the Merchant, the Spender never needs to provide any identification. This is the same as for purchases made with conventional cash. Also, note that the Bank never sees the values of A, B, z, a, b, r for the coin until it is deposited by the Merchant. In fact, the Bank provides only the number w and the number c_1, and has seen only c. However, the coin still contains information that identifies the Spender in the case of double spending. Is it possible for the Merchant or the Bank to extract the Spender's identity from knowledge of the coin (A, B, z, a, b, r) and the triple (r_1, r_2, d)? Since the Bank also knows the identification number I, it suffices to consider the case where the Bank is trying to identify the Spender. Since s, x_1, x_2 are secret random numbers known only to the Spender, A and B are random numbers. In particular, A is a random power of g and cannot be used to deduce I. The number z is simply $A^x \pmod p$, and so does not provide any help beyond what is known from A. Since a and b introduce two new secret random exponents α_1, α_2, they are again random numbers from the viewpoint of everyone except the Spender.

At this point, there are five numbers, A, B, z, a, b, that look like random powers of g to everyone except the Spender. However, when $c \equiv \alpha_1^{-1} H(A, B, z, a, b) \pmod q$ is sent to the Bank, the Bank might try to compute the value of H and thus deduce α_1. But the Bank has not seen the coin and so cannot compute H. The Bank could try to keep a list of all values c it has received, along with values of H for every coin that is deposited, and then try all combinations to find α_1. But it is easily seen that, in a system with millions of coins, the number of possible values of α_1 is too large for this to be practical. Therefore, it is unlikely that knowledge of c, hence of b, will help the Bank identify the Spender.

The numbers α_1 and α_2 provide what Brands calls a **restricted blind signature** for the coin. Namely, using the coin once does not allow identification of the signer (namely, the Spender), but using it twice does (and the Spender is sent to jail, as pointed out previously).

To see the effect of the restricted blind signature, suppose α_1 is essentially removed from the process by taking $\alpha_1 = 1$. Then the Bank could keep a list of values of c, along with the person corresponding to each c. When a coin is deposited, the value of H would then be computed and compared with the list. Probably there would be only one person for a given c, so the Bank could identify the Spender.

16.3 Bitcoin Overview

In this section we provide a brief overview of Bitcoin. For those interested in the broader issues behind the design of cryptocurrencies like Bitcoin, we refer to the next section.

Bitcoin is an example of a ledger-based cryptocurrency that uses a combination of cryptography and decentralized consensus to keep track of all of the transactions related to the creation and exchange of virtual coins, known as bitcoins.

Bitcoin is a very sophisticated collection of cryptography and communication protocols, but the basic structure behind the operation of Bitcoin can be summarized as having five main stages:

- Users maintain a transaction ledger;
- Users make transactions and announce their transactions;
- Users gather transactions into blocks;
- Users solve cryptographic puzzles using these blocks;
- Users distribute their solved puzzle block.

Let’s start in the middle. There are many users. Transactions (for example, Alice gives three coins to Bob and five coins to Carla) are happening everywhere. Each transaction is broadcast to the network. Each user collects these transactions and verifies that they are legitimate. Each user collects the valid transactions into a block. Suddenly, one user, say Zeno, gets lucky (see “Mining” below). That user broadcasts this news to the network and gets to add his block to the ledger that records all transactions that have ever taken place. The transactions continue throughout the world and continue to be broadcast to everyone. The users add the valid transactions to their blocks, possibly including earlier transactions that were not included in the block that just got added to the ledger. After approximately 10 minutes, another user, let's say Xenia, gets lucky and is allowed to add her block to the ledger. If Xenia believes that all of the transactions are valid in the block that Zeno added, then Xenia adds her block to the ledger that includes Zeno's block. If not, then Xenia adds her block to the ledger that was in place before Zeno's block was added. In either case, Xenia broadcasts what she did.

Eventually, after approximately another 10 minutes, Wesley gets lucky and gets to add his block to the ledger. But what if there are two or more competing branches of the ledger? If Wesley believes that one contains invalid transactions, he does not add to it, and instead chooses among the remaining branches. But if everything is valid, then Wesley chooses the longest branch. In this way, the network builds a consensus as to the validity of transactions. The longer branch has had more randomly chosen users certify the transactions it contains.

Stopping Double Spending. Now, suppose Eve buys something from the vendor Venus and uses the same coins to buy something from the seller Selena. Eve broadcasts two transactions, one saying that she paid the coins to Venus and one saying that she paid the coins to Selena. Some users might add one of the transactions to their blocks, and some add the other transaction to their blocks. There is possibly no way someone can tell which is legitimate. But eventually, say, Venus's block ends up in a branch of blocks that is longer than the branch containing Selena's block. Since this branch is longer, it keeps being augmented by new blocks, and the payment to Selena becomes worthless (the other transactions in the block could be included in later additions to the longer branch). What happens to Selena? Has she been cheated? No. After concluding the deal with Eve, Selena waits an hour before delivering the product. By that time, either her payment has been included in the longer branch, or she realizes that Eve's payment to her is worthless, so Selena does not deliver the product.

Incentives. Whenever a user gets lucky and is chosen to add a block to the ledger, that user collects fees from each transaction that is included in the block. These payments of fees are listed as transactions that form part of the block that is added to the ledger. If the user includes invalid transactions, then it is likely that a new branch will soon be started that does not include this block, and thereby the payments of transaction fees become worthless. So there is an incentive to include many transactions, but there is also an incentive to verify their validity. At present, there is also a reward for being the lucky user, and this is included as a payment in the user's block that is being added to the ledger. After every 210000 blocks are added to the ledger, which takes around four years at 10 minutes per block, the reward amount is halved. In 2018, the reward stood at 25 bitcoins. The overall system is set up so that there will eventually be a total of 21 million bitcoins in the system. After that, the plan is to maintain these 21 million coins as the only bitcoins, with no more being produced. At that point, the transaction fees are expected to provide enough incentive to keep the system running.

Mining. How is the lucky user chosen? Each user computes

$$h(\text{Nonce}||\text{prevhash}||\text{Trans}X_1||\text{Trans}X_2|\cdots||\text{Trans}X_N)$$

for billions of values of *Nonce*. Here, h is the hash function SHA-256, *Nonce* is a random bitstring to be found, *prevhash* is the hash of the previous block in the blockchain, $TransX_j$ are the transactions that the user is proposing to add to the ledger. On the average, after around 10^{20} hashes are computed worldwide, some user obtains a hash value whose first 66 binary digits are 0s.

(These numbers are adjusted from time to time as more users join in order to keep the average spacing at 10 minutes.) This user is the “lucky” one. The nonce that produced the desired hash is broadcast, along with the hash value obtained and the block that is being added to the ledger. The mining then resumes with the updated ledger, at least by users who deem the new block to be valid.

Mining uses enormous amounts of electricity and can be done profitably only when inexpensive electric power is available. When this happens, massive banks of computers are used to compute hash values. The rate of success is directly proportional to the percentage of computer power one user has in relation to the total power of all the users in the world. As long as one party does not have access to a large fraction of the total computing power, the choice of the lucky user will tend to be random enough to prevent cheating by a powerful user.

16.3.1 Some More Details

Users Maintain a Transaction Ledger: The basic structure behind Bitcoin is similar to many of the other ledger-based cryptocurrencies. No actual *digital coins* are actually exchanged. Rather, a ledger is used to keep track of transactions that take place, and pieces of the ledger are the digital objects that are shared. Each user maintains their own copy of the ledger, which they use to record the community’s collection of transactions. The ledger consists of blocks, structured as a blockchain (see Section 12.7), which are cryptographically signed and which reference previous blocks in the ledger using hash pointers to the previous block in the ledger.

Also, a transaction includes another hash pointer, one to the transaction that says that the spender has the bitcoins that are being spent. This means that when someone else checks the validity of a transaction, it is necessary to look at only the transactions that occurred since that earlier transaction. For example, if George posts a transaction on June 14 where he gives 13 bitcoins to Betsy, George includes a pointer to the transaction on the previous July 4 where Tom paid 18 bitcoins to George. When Alex wants to check the validity of this transaction, he checks only those transactions from July 4 until June 14 to be sure that George didn't also give the bitcoins to Aaron. Moreover, George also can post a transaction on June 14 that gives the other five bitcoins from July 4 to himself. In that way, the ledger is updated in a way that there aren't small pieces of long-ago transactions lying around.

Making and Announcing Transactions: A transaction specifies the coins from a previous transaction that are being consumed as input. As output of the transaction, it specifies the address of the recipients and the amount of coins to be delivered to these recipients. For example, in addresses for Bitcoin Version 1, each user has a public/private pair of keys for the Elliptic Curve Digital Signature Algorithm. The user's address is a 160-bit cryptographic hash of their public key. The 160-bit hash is determined by first calculating the SHA-256 hash of the user's public key, and then calculating the RIPEMD-160 hash (this is another widely used hash function) of the SHA-256 output. A four-byte cryptographic checksum is added to the 160-bit hash, which is calculated from SHA-256 being applied twice to the 160-bit hash. Finally, this is encoded into an alphanumeric representation.

The transaction is signed by the originator of the transaction using their private key, and finally it is announced to the entire set of users so it can be added to blocks that will be appended to the community's ledger.

Gathering Transactions into Blocks: Transactions are received by users. Users verify the transactions that they receive and discard any that they are not able to verify. There are several reasons why transactions might not verify. For example, since the communications are taking place on a network, it is possible that not every user will receive the same set of transactions at the same time. Or, users might be malicious and attempt to announce false transactions, and thus Bitcoin relies on users to examine the collection of new transactions and previous transactions to ensure that no malicious behavior is taking place (such as double spending, or attempting to steal coins). Users then gather the transactions they believe are valid into the block that they are forming. A new, candidate block consists of a collection of transactions that a user believes is valid, and these transactions are arranged in a Merkle Tree to allow for efficient searching of transactions within a block. The block also contains a hash pointer to a previous block in the ledger. The hash pointer is calculated using the SHA-256 hash of a previous block.

Anonymity: Any cash system should have some form of anonymity. In Bitcoin, a user is identified only through that user's public key, not through the actual identity. A single user could register under multiple names so that an observer will not see several transactions being made by one user and deduce the user's identity. Of course, some anonymity is lost because this user will probably need to make transfers from the account of one of his names to the account of another of his names, so a long-term analysis might reveal information.

An interesting feature of Bitcoin is that registering under multiple names does not give a user more power in the mining operations because the computational resources for mining will be spread over several accounts but will not increase in power. Therefore, the full set of this user's names has the same probability of being lucky as if the user had opened only a single account. This is much better than the alternative where a consensus could be reached by voting with one vote per account.

16.4 Cryptocurrencies

In this section, we give a general discussion of issues related to digital cash, using Bitcoin as an example.

The Brands digital cash system from Section 16.2 gives insight into how difficult it can be to re-create the benefits of cash using just cryptography and, even with all its cryptographic sophistry, the Brands scheme is still not able to allow for the cash to be transferred to others or for the digital money to be divided into smaller amounts. Up until the 2000s, almost all digital cash protocols faced many shortcomings when it came to creating digital objects that acted like real-world money. For example, in order to allow for anonymity and to prevent double spending, it was necessary to introduce significant complexity into the scheme. Some protocols were able to work only online (i.e., they required the ability to connect to an agent acting as a bank to verify the validity of the digital currency), some protocols like Brands's system did not allow users to break cash into smaller denominations and thereby issue change in transactions. Beyond the technical challenges, some systems, like Chaum's ECash, faced pragmatic hurdles in persuading banks and merchants to adopt their system. Of course, without merchants to buy things from, there won't be users to make purchases, and this can be the death of a cryptocurrency.

One of the major, practical challenges that the early forms of cryptocurrencies faced was the valuation of their digital cash. For example, just because a digital coin is supposedly worth $100, what actually makes it worth that amount? Many early digital currencies attempted to tie themselves to real-world currencies or to real-world commodities (like gold). While this might seem like a good idea, it also introduced several practical challenges: it implicitly meant that the value of the entire cryptocurrency had to be backed by an equivalent amount of real-world cash or coins. And this was a problem – if these new, digital currencies wanted to exist and be used, then they had to acquire a lot of real-world money to back them.

This also meant that the new forms of digital cash weren't actually their own currency, but actually just another way to spend an existing currency or commodity. Since needing real-world assets in order to get a cryptocurrency off the ground was a hurdle, it led many to ask "What if one could make the digital cash its own currency that was somehow valued independently from other currencies?"

It turns out that many of the technical solutions needed to overcome the various hurdles that we have outlined already existed in the early 2000s and what was needed was a different perspective. In 2008, Satoshi Nakamoto presented a landmark paper [Nakamoto], which launched the next generation of cryptocurrencies and introduced the well known cryptocurrency Bitcoin. These cryptocurrencies, which include Bitcoin, Ethereum, and many others, have been able to overcome many of the hurdles that stifled previous attempts such as ECash. Currently, currencies like Bitcoin and Ethereum are valid currencies that are traded internationally.

The new generation of cryptocurrencies were successful because they made practical compromises or, to put it another way, they did not try to force all of the properties of real-world cash onto digital currencies. For example, a digital currency like Bitcoin does not work offline, but it does cleverly use decentralization to provide robustness so that if parts of the system are offline, the rest of the system can continue to function. Another major paradigm shift was the realization that one did not need to create a digital object corresponding to a coin or to cash, but rather that the money could be *virtual* and one only needed to keep track of the trading and exchange of this virtual money. That is, bookkeeping was what was important, not the coin itself! And, once we stop trying to create a digital object that acts like cash, then it becomes a lot easier to achieve properties like preventing double spending and being able to divide coins into smaller parts. Another paradigm shift was the realization that, rather than tying the value of the currency to real-world currency, one could tie the value to solving a hard (but not impossible) computational problem and make the solution of that computational problem have value in the new currency.

In order to understand how these new cryptocurrencies work, let’s take a look at the basic structures being used in Bitcoin. Many of the other currencies use similar building blocks for their design.

As we just mentioned, one of the major changes between Bitcoin and older digital cash schemes is a move away from a digital object representing a coin and instead to the use of an imaginary or virtual currency that is kept track of in bookkeeping records. These ledgers, as they are known, record the creation and exchange of these virtual coins. The use of cryptography to form secure ledgers has become a growing trend that has impacted many applications beyond the support of cryptocurrencies.

Let us look at a simple ledger. To start, assume that we have a central bank, which we call BigBank. Later we shall remove BigBank, but for now it is useful to have BigBank around in order to start the discussion. We suppose that BigBank can make new virtual coins, and that it wants to share them with Alice. To do so, it makes a ledger that records these events, something that looks like:

Ledger
BigBank: Create 100 coins
BigBank $\rightarrow$ Alice: 100 coins

This ledger can be shared with Alice or anyone else, and after looking at it, one can easily figure out the sequence of events that happened. BigBank made 100 coins, and then these coins were given by BigBank to Alice. Assuming there have not been any other transactions, then one can conclude that Alice is now the owner of 100 virtual coins.

Of course, Alice should not just trust this ledger. Anyone could have made this ledger, pretending to be BigBank. As it is, there is nothing tying this ledger to its creator. To solve this problem, we must use digital signatures. Therefore, BigBank has a public–private key pair and has shared its public key with Alice and the rest of the world. There are many ways in which this could have been done, but it is simplest to think of some certificate authority issuing a certificate containing BigBank's public key credentials. Now, rather than share only the Ledger, BigBank also shares $\text{sign}_{BB}(\text{Ledger})$.

This makes it harder for someone to pretend to be BigBank, but we still have some of the usual problems that we encountered in Chapter 15. For example, one problem that stands out is that it is possible to perform a replay attack. If Alice receives five separate copies of the signed Ledger, then she might conclude that she has 500 coins. Therefore, in order to fix this we need to use some form of unique transaction identifier or counter in Ledger before BigBank signs it, something like:

	Ledger 1
TID-1.	BigBank: Create 100 coins
TID-2.	BigBank $\rightarrow$ Alice: 100 coins

Now, the signed Ledger allows anyone to have some trust that BigBank has given Alice 100 coins.

Alice’s coins aren’t much use to her unless she can spend them. Suppose she goes to Sarah’s Store, which is known to accept BigBank’s coins, and wants to buy an item using BigBank’s coins. Alice could write another entry in the ledger by giving Sarah’s Store an update to the ledger that looks like

$$\begin{array}{c} \text{Ledger 2} \\ \text{TID-3.} \quad \text{Alice} \to \text{SarahStore: 100 coins} \end{array}$$

Now, Alice signs this update and sends Sarah’s Store $\text{sign}_{Alice}(\text{Ledger } 2)$. Sarah’s Store can indeed verify that Alice made this update, but a problem arises: How does Sarah’s Store know that Alice in fact had 100 coins to spend?

A simple way to take care of this is for Alice to attach a signed copy of BigBank’s original Ledger. Now Sarah’s Store can verify that Alice did get 100 coins from BigBank.

This works, but we come to classic double spending problem that all digital currencies must address. Right now, there is nothing preventing Alice from going to Vendor Veronica and spending those 100 coins again. Alice can simply make another version of Ledger 2, call it Ledger $2'$:

$$\begin{array}{c} \text{Ledger } 2' \\ \text{TID-3.} \quad \text{Alice} \to \text{Vendor Veronica: 100 coins} \end{array}$$

She can sign Ledger $2'$ and send both it and BigBank’s signed copy of the original Ledger 1. But even with these, there is no way for Vendor Veronica to be assured that Alice did not spend her 100 coins elsewhere, which she in fact did.

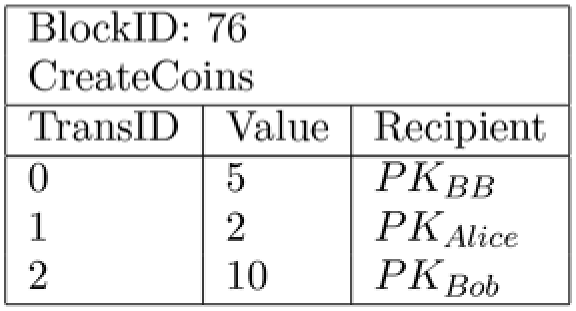
What is needed is a way to solve the double spending problem. We can do this by slightly revising the purchasing protocol to place BigBank in a more central role. Rather than let Alice announce her transaction, we require Alice to contact BigBank when she makes purchases and BigBank to announce updates to the ledger. The ledger of transactions operates as an append-only ledger in that the original ledger with all of its updates contains the history of all transactions that have ever occurred using BigBank's coins. Since it is not possible to remove transactions from the ledger, it is possible for everyone to see if double spending occurs simply by keeping track of the ledger announced by BigBank.

Ensuring that this append-only ledger is cryptographically protected requires that each update is somehow tied to previous versions of the ledger updates, otherwise it might be possible for transactions to be altered. The construction of the append-only ledger is built using a cryptographic data structure known as a blockchain, which we introduced in Section 12.7. A blockchain consists of two different forms of hash-based data structures. First, it uses hash pointers in a hash chain, and second it uses a Merkle tree to store transactions. The use of the hash chain in the blockchain makes the ledger tamper-proof. If an adversary attempts to tamper with data that is in block k, then the hash contained in block $k+1$, namely, the hash of the correct block k, will not match the hash of the altered block k. The use of the Merkle tree provides an efficient means of determining whether a transaction belongs to a block.

Now suppose that BigBank wants to keep track of all of the transactions that have taken place and publish it so that others can know these transactions. BigBank could periodically publish blocks, or it could gather an entire collection of blocks and publish them all at once. Ultimately, BigBank publishes all of the blocks of the blockchain along with a final hash pointer that certifies the entire collection of blocks in the blockchain. Each of the blocks in the blockchain, along with the final hash pointer, is signed by BigBank so that others know the identity of the entity publishing the ledger. Thus, the hash chaining in the blockchain provides data integrity, while the digital signatures applied to each block provide origin authentication.

Let us return to the story of BigBank and all of the various participants who might want to use or acquire some of BigBank’s digital coins. Everyone wishing to use some of their BigBank coins must tell BigBank how many coins or how much value is being transferred for a purchase and whom the coins will be given to. BigBank does not need to know what is being purchased, or why there is a transaction occurring; it just needs to know the value of the exchange and the recipient. BigBank will then gather several of these individual transactions into a single block, and publish that block as an update to the blockchain that represents BigBank’s ledger.

To do this, though, BigBank needs a way to describe the transactions in the ledger, and this means that there needs to be a set of basic transaction types that are allowed. It turns out that we don’t need very many to have a useful system. For example, it is useful for an entity like BigBank to create coins, where each coin has a value, a serial number, and a recipient. BigBank can, in fact, create as many coins as it wants during the time before it publishes the next block in the blockchain. For example, suppose BigBank creates three coins with different recipients, and that it publishes the creation of these coins in the data portion of a block that looks like:



BlockID: 76 CreateCoins		
TransID	Value	Recipient
0	5	PK_{BB}
1	2	PK_{Alice}
2	10	PK_{Bob}

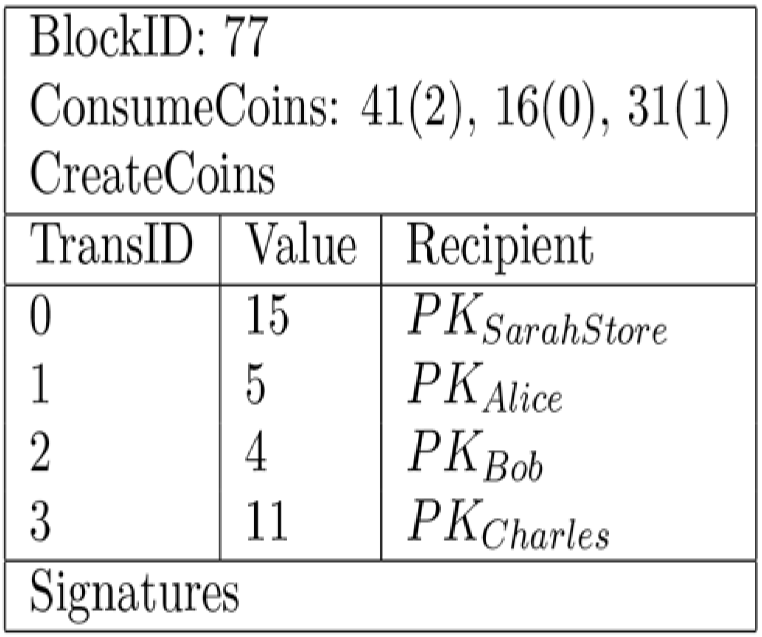
16.4-1 Full Alternative Text

When this block is published, it informs the rest of the world that BigBank created several coins of different values. First, in transaction 0 for block 76, a coin worth 5 units was created and given to PK_{BB} (which, as will be explained shortly, stands for BigBank's Public Key). This coin will be referred to as coin 76(0) since it was made in block 76, transaction 0. Similarly, a coin was created in transaction 1 of Block 76 for PK_{Alice}, which will be referred to as coin 76(1). Lastly, a coin was created in transaction 2 for PK_{Bob}, and will be referred to as coin 76(2). Who or what are PK_{BB}, PK_{Alice}, and PK_{Bob}? This is an example of something clever that Bitcoin borrowed from other applications of cryptography. In [HIP], it was recognized that the notion of one's identity is best tied to something that only that entity should possess. This is precisely what a private key provides in public key cryptography, and so if someone wants to send something to Alice or Bob, then they can use Alice or Bob's public key (which they know because the public key is public) as the name of the recipient.

In order to support purchases made by Alice and others, we need to allow for the users of BigBank currency to consume coins in exchange for goods, and to receive change for their transactions. It was realized that it was easier to simply destroy the original coin and issue new coins as payment to the vendors and new coins as change to those making purchases than it was to break a coin down into smaller denominations, which had proven difficult to accomplish for previous digital coin schemes. So, if Bob has a coin 76(2) worth 10 units, and he wants to buy something worth 7 units, BigBank destroys 76(2), issues a new coin to the vendor worth 7 units, and issues a new coin to Bob worth 3 units.

Of course, BigBank is big and might need to handle many transactions during the time it takes to form a block. Therefore, BigBank needs to hear from all of its customers about their purchases, and it needs to make certain that the coins that its customers are spending are valid and not previously spent. It also needs to create new coins and give the new coins to those who are selling to BigBank's customers, while also issuing new coins as change. Let us look at how this could work. Suppose Alice has a coin 41(2) that is worth 20 units, and that she wants to buy a widget from Sarah's Store for 15 units. Then, the first step that BigBank takes is to destroy Alice's coin worth 20 units, then issue a new coin to Sarah's Store worth 15 units and a new coin to Alice worth 5 units, which corresponds to Alice's change from her purchase. Thus, one coin was consumed and two were created, but the total value of the coins being consumed is equal to the total value of the coins being created from the transaction. Lastly, before BigBank can publish its update to the ledger, it needs to get all of the owners of the coins that were consumed to sign the transaction, thereby ensuring that they knowingly spent their coins. Any good cryptographic signature scheme would work. Bitcoin uses the Elliptic Curve Digital Signature Algorithm (ECDSA), which we will see in Exercise 24 in Chapter 21.

The next block of transactions in the blockchain thus looks like:



BlockID: 77
ConsumeCoins: 41(2), 16(0), 31(1)
CreateCoins

TransID	Value	Recipient
0	15	$PK_{SarahStore}$
1	5	PK_{Alice}
2	4	PK_{Bob}
3	11	$PK_{Charles}$

Signatures

16.4-2 Full Alternative Text

BigBank's approach to managing its ledger of transactions seems to work well, but it has one major problem, and that is the fact that BigBank has to be involved in every transaction. Not only does this mean a lot of work for BigBank, but it also leads to several security concerns, such as the risk of a denial of service to the system if BigBank becomes disconnected from the rest of the network, or the risk that BigBank might use its influence as the central entity in the economy to extract extortion from its members. Although BigBank cannot create false transactions, it could ask for large *service* fees in order to carry out a transaction.

Bitcoin gets around the problem of a central bank vetting each and every transaction by making the blockchain protocol decentralized. In other words, rather than having a single entity check the transactions, Bitcoin uses the notion of community consensus to check that the transactions are correct. Also, rather than have a single entity maintain and update the ledger of transactions, Bitcoin allows the members of the community to form the next block in the blockchain and thereby update the ledger themselves. In fact, Bitcoin went further in that it did not trust banks to mint coins themselves. Instead, Bitcoin devised a means by which coins get created periodically and are awarded to those who do work to keep the Bitcoin system moving smoothly.

In order to understand this, we need to set the stage for how life in the Bitcoin universe works. Every T units of time (in Bitcoin, $T = 10 \text{ minutes}$), the users in the Bitcoin network must agree on which transactions were broadcast by the users making purchases and trading in bitcoins. Also, the nodes must agree on the order of the transactions, which is particularly important when money is being exchanged for payments. Bitcoin then proceeds in rounds, with a new update to the ledger being published each round.

Every user in the Bitcoin network keeps track of a ledger that contains all of the blocks that they've agreed upon. Every user also keeps a list of transactions that they have heard about but have not yet been added to the blockchain. In particular each user might have slightly different versions of this list, and this might happen for a variety of reasons. For example, one user might have heard about a transaction because it is located near that transaction, while another user might be further away and the announcement of the transaction might not have made it across the network to that user yet. Or, there might be malicious users that have announced incorrect transactions and are trying to put wrong transactions into the blockchain.

Bitcoin uses a consensus protocol to work out which transactions are most likely correct and which ones are invalid. A consensus protocol is essentially a way to tally up how many users have seen and believe in a transaction. In each round of the consensus protocol, new transactions are broadcast to all of the users in the Bitcoin network. These transactions must be signed by those spending the coins involved. Each user collects new transactions and puts the transactions it believes into a new block. In each round, a *random* user will get to announce its block to all of the other users. Bitcoin uses a clever way to determine which user will be this random user, as we shall soon discuss. All of the users that get this new block then check it over and, if they accept it, they include its hash in the next block that they create. Bitcoin uses the SHA-256 hash function for hashing the block and forming the hash pointer.

Before we proceed to discuss how Bitcoin randomly chooses which user will be chosen to announce its block, we look at some security concerns with this protocol. First is the concern that Alice could attempt to double spend a coin. In each round, coins are being created and destroyed, and this is being logged into the ledger. Since the ledger is being announced to the broader community, other users know which coins have been spent previously and thus they will not accept a transaction where Alice tries to spend a coin she already used. Much of the trustworthiness of Bitcoin's system is derived from the fact that a consensus protocol will arrive at what most users believe is correct and, as long as most users are honest, things will work as they should with false transactions being filtered before they get into the blockchain.

Another concern is that Alice could write transactions to give herself coins from someone else. This is not possible because, in order to do so, Alice would have to write a transaction as if she was another user, and this would require her to create a valid signature for another user, which she is unable to do.

A different concern is whether Alice could prevent a transaction from Bob from being added to the ledger. Although she could choose not to include Bob's transaction in a block she is creating, there is a good chance that one of the other users in the network will announce a block with Bob's transaction because the user that is chosen to announce its block is randomly chosen. Additionally, even if she is chosen to announce her block, a different honest user will likely be chosen during the next round and will simply include Bob's transaction. In Bitcoin, there isn't any problem with a transaction taking a few rounds to get into the blockchain.

We now return to the question of how nodes are randomly selected. Remember that we needed to remove the role of a central entity like BigBank, and this means that we cannot rely on a central entity to determine which user will be randomly selected. Bitcoin needed a way to choose a user randomly without anyone controlling who that user would be. At the same time, we can't just allow users to decide selfishly that they are the one chosen to announce the next block – that could lead to situations where a malicious user purposely leaves another user's transactions out of the blockchain.

Bitcoin also needed a way to encourage users to want to participate by being the random node that announces a new block. The design of Bitcoin allows users who create blocks to be given a reward for making a block. This reward is known as a block reward, and is a fixed amount defined by the Bitcoin system. Additionally, when a user engages in a transaction that it wants logged into the blockchain ledger, it can set aside a small amount of value from the transaction (taking a little bit from the change it might issue itself, for example), and give this amount as a service charge to the user that creates and adds the block containing the transaction into the blockchain.

Together, these two incentives encourage users to want to make the block, so the challenge then is ensuring that random users are selected. Bitcoin achieves this by requiring that users perform a task where it isn't guaranteed which user will be the first to complete the task. The first user to complete the task will have performed the *proof of work* needed to be randomly selected. Users compete with each other using their computing power, and the likelihood that a user will be selected ”randomly” depends on how much computing power they have. This means that if Alice has twice the computing power that Bob has, then she will be twice as likely to be the first to complete the computing task. But, even with this advantage, Alice does not control whether she will finish the task first, and so Bob and other users have a chance to publish the block before she does.

Bitcoin’s proof of work requires that users solve a hash puzzle in order to be able to announce a block to be added to the blockchain. Specifically, a user who wishes to propose a block for adding the community’s ledger is required to find a nonce such that

$$h(Nonce \parallel prevhash \parallel TransX_1 \parallel TransX_2 \parallel \cdots \parallel TransX_N) < B,$$

where $Nonce$ is the nonce to be found, $prevhash$ is the hash of the previous block in the blockchain, $\{TransX_j\}$ are the transactions that the user is proposing to add to the ledger, and B is a threshold value for the hash puzzle. What this means is the hash value is interpreted as an integer in binary and there must be a certain number of 0s at the beginning of this binary expansion. The value of B is chosen so that it will take roughly T units of time for some user to solve the puzzle. In Bitcoin, $T = 10 \text{ minutes}$, and every two weeks the Bitcoin protocol adjusts B so that the average time needed to solve a hash puzzle remains around 10 minutes. Bitcoin uses the SHA-256 hashing algorithm twice for the proof of work cryptopuzzle, once for the hash of the previous ledger and once for obtaining the desired small hash value. In practice, the user does a massive computation using numerous nonces, hoping to be the first to find an appropriate hash value.

Once a user finds this nonce, it forms the new block, which includes $Nonce$, the previous hash, and the transactions it has included. Then the user announces the block to the entire community. The reward for completing this task first is that the user will receive the block reward, which is several bitcoins, as well as transaction fees. The process of solving the hash puzzle and announcing a block is called Bitcoin mining, and has become a popular activity with many companies devoting large amounts of computing (and electrical power) to solving puzzles and reaping the Bitcoin rewards.

Let's now put everything together and describe what a typical Bitcoin blockchain looks like. A block in Bitcoin's blockchain consists of a header followed by a collection of transactions that have been organized as a Merkle tree. The header itself contains several different pieces of information: the hash pointer to the previous block in the blockchain, a timestamp, the root of the Merkle tree, and the nonce. In actual implementation of Bitcoin, it is only the hash of the header that must satisfy the conditions of the cryptographic puzzle, rather than the hash of the full block of information. This makes it easier to verify a chain of blocks since one only needs to examine the hash of the headers, and not the hashes of many, many transactions. We may therefore visualize the Bitcoin blockchain as illustrated in Figure 16.1.

Figure 16.1 Each block in Bitcoin's blockchain consists of a header and a Merkle tree of transactions

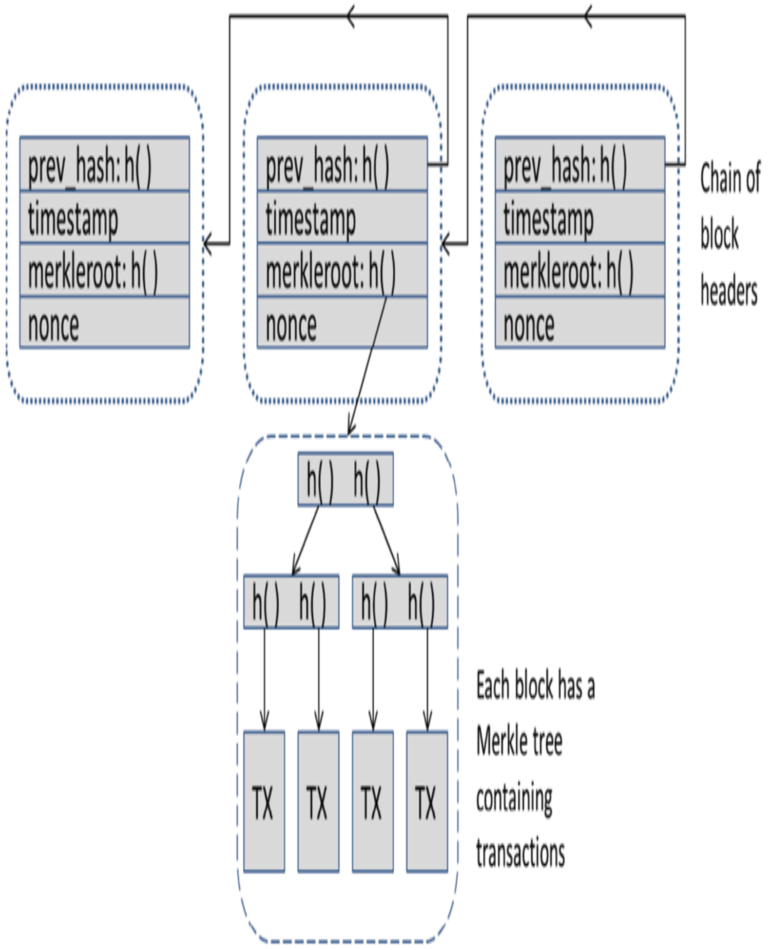


Figure 16.1 Full Alternative Text

Our description of Bitcoin is meant to convey the ideas behind cryptocurrencies in general and thus greatly simplifies the details of the actual Bitcoin protocol. We have avoided describing many practical matters, such as the formal specification of the language used by Bitcoin to specify transactions, the data structures and housekeeping that are needed to keep track of transactions, as well as the details behind the use of the hash of a public key for addressing users. The reader can look at [Bitcoin] or [Narayanan et al.] to find the details behind Bitcoin and its protocols.

16.5 Exercises

1. In the scheme of Section 16.2, show that a valid coin satisfies the verification equations

   $$g^r \equiv a\, h^{H(A,\, B,\, z,\, a,\, b)},\;\; A^r \equiv z^{H(A,\, B,\, z,\, a,\, b,\, r)} b,\;\; \text{and } g_1^{r_1} g_2^{r_2} \equiv A^d B \pmod{p}.$$

2. In the scheme of Section 16.2, a hacker discovers the Bank's secret number x. Show how coins can be produced and spent without having an account at the bank.

3. In the scheme of Section 16.2, the numbers g_1 and g_2 are powers of g, but the exponents are supposed to be hard to find. Suppose we take $g_1 = g_2$.

   1. Show that if the Spender replaces r_1, r_2 with $r_1^{'}, r_2^{'}$ such that $r_1 + r_2 = r_1' + r_2'$, then the verification equations still work.

   2. Show how the Spender can double spend without being identified.

4. Suppose that, in the scheme of Section 16.2, the coin is represented only as (A, B, a, r); for example, by ignoring z and b, taking the hash function H to be a function of only A, B, a, and ignoring the verification equation $A^r \equiv z^H b$. Show that the Spender can change the value of u to any desired number (without informing the Bank), compute a new value of I, and produce a coin that will pass the two remaining verification equations.

5. In the scheme of Section 16.2, if the Spender double spends, once with the Merchant and once with the Vendor, why is it very likely that $r_2 - r_2^{'} \not\equiv 0 \pmod{q}$ (where $r_2, r_2^{'}$ are as in the discussion of Fraud Control)?

6. A Sybil attack is one in which an entity claims multiple identities or roles in order to achieve an advantage against a system or protocol. Explain why there is no advantage in launching a Sybil attack against Bitcoin's proof of work approach to determining which user will randomly be selected to announce the next block in a blockchain.

7. Forgetful Fred would like to save a file containing his homework to a server in the network, which will allow him to download it later when he needs it. Describe an approach using hash functions that will allow Forgetful Fred to verify that he has obtained the correct file from the server, without requiring that Fred keep a copy of the entire file to check against the downloaded file.

8. A cryptographic hash puzzle involves finding a nonce n that, when combined with a message m, will hash to an output y that belongs to a target set S, i.e. $h(n \parallel x) \in S$.

    1. Assuming that all outputs of a $k\text{-bit}$ cryptographic hash function are equally likely, find an expression for the average amount of nonces n that need to be tried in order to satisfy

    $$h(n \parallel x) < 2^L.$$

    2. Using your answer from part (a), estimate how many hashes it takes to obtain a hash value where the first 66 binary digits are 0s.