Editors: Francis Sullivan, fran@super.org | Ernst Mucke, ernst.mucke@gmail.com

What Is the Blockchain?

Massimo Di Pierro | DePaul University

The technology known as the blockchain was first revealed by Satoshi Nakamoto in his paper "Bitcoin: A Peer-to-Peer Electronic Cash System" (https://bitcoin.org/bitcoin.pdf), which laid out the mathematical foundation for the bitcoin cryptocurrency. Although this was a groundbreaking paper, it was never actually submitted to a traditional peer-reviewed journal, and the author's true identity is unknown. Blockchain technology is not only at the foundation of all crytocurrencies, but it has found wide application in the more traditional financial industry. It also opened the door to new applications such as smart contracts.

It's a Matter of Trust

The problem that Nakamoto solved with the blockchain was that of establishing trust in a distributed system. More specifically, the problem of creating a distributed storage of timestamped documents where no party can tamper with the content of the data or the timestamps without detection.

Note that this problem is orthogonal to the problems of authentication, integrity, and nonrepudiation, which are solved by digital signatures. If a party creates a digital signature for a document, it establishes only a verifiable link between the party and the document. The existence of a valid digital signature proves that the party indeed intended to sign the document and that the document hasn't been altered. Yet the digital signature guarantees nothing about the time when the document was signed: the timestamp requires trust in the party that signed it. In the case of financial transactions and other forms of legal contracts, time is of the essence, and the order of those financial transactions needs to be independently certified to be auditable.

Consider the case of house sales. The owner can be defined as the party to whom the house was last sold to, but ownership can only be verified from the full paper trail of all transactions related to the house, a paper trail that's

 1521-9615/17/$33.00 © 2017 IEEE Copublished by the IEEE CS and the AIP

usually kept and verified by title companies. Note this system doesn't completely prevent fraudulent transactions (such as a person selling a house that he or she doesn't own or selling the same property to more than one party), but fraudulent activities eventually get detected, and true ownership is established. The same ownership verification problem arises in financial transactions—for sure, in the sale of cryptocurrency, but also in the sale of any other traditional financial instrument. The problem is normally solved by recording all transactions in a single trusted centralized ledger, but a ledger isn't always a practical solution because it doesn't scale to large numbers of frequent transactions and because it requires all parties to trust the ledger's maintainer. In the same way you need to trust your bank with your money (and bank employees stealing customer funds is not unheard of). To address this, the blockchain provides a distributed trust mechanism: multiple parties keep a record of transactions, and every party can verify that the order and timestamps of the transactions haven't been tampered with.

A unit of bitcoin is nothing other than a number, but only some numbers are valid bitcoins. These numbers are solutions of a well-defined equation, and whoever finds a new solution owns it (this process is called mining). Once a bitcoin is discovered, it can be traded, with transactions stored in a ledger. Transactions are digitally signed with the credentials of the seller to avoid nonrepudiation. There is no centralized ledger because users wouldn't trust one and because there are too many transactions to store them all in one place. Hence bitcoin and other cryptocurrencies provide a distributed ledger in which every computer involved in the transaction of a specific coin (or fraction of a coin) keeps a copy of the history of that coin's transactions. The blockchain technology makes sure that no party storing this history can tamper with it without being detected.

Hash Functions

Transactions are units of data containing the transaction details plus a timestamp. Both can be represented as computer numbers or strings. A blockchain can be thought of as a table with three columns, where each row represents a distinct transaction, the first column stores the transaction's timestamp, the second column stores the transaction's details, and the third column stores a hash of the current transaction plus its details plus the hash of the previous transaction. When a new record is inserted into a blockchain, the last computed hash is broadcasted to every interested party. It isn't necessary for every party to keep a copy of the entire transaction history—it's sufficient that a few parties do. Because everyone knows the last hash, anyone can verify that the data hasn't been altered since it would be impossible without obtaining a different and thus invalid hash. The only way to tamper with the data while preserving the hash would be to find a collision in the data, and that's computationally impossible. It would require so much computing power that it's practically uneconomical.

A hash can be thought of as an encrypted version of the original string from which it is impossible to derive the original string. In fact, one way to compute the hash of a string is by encrypting it and performing some scrambling and xoring of the output bits. Mathematically, a hash is produced by a hash function, f, which must have two important properties: the size of the input space and the output space must be large; it must be practically impossible to find collisions, that is, two inputs x_1 and x_2 that produce the same output $f(x_1) = f(x_2)$. A typical application of hash functions is in password storage—when you register on a website, you don't want the site to store your password p in its database, otherwise anyone with access to the database could read it. The website should store the hash of the password, $f(p) = y$. When you login, the input password p is hashed again and compared with the stored value, $f(p) = y$. The probability of an incorrect password producing the same hash value y as the actual password is zero for practical purposes.

Examples of hash functions are the Secure Hash Algorithms (SHA1, SHA128, SHA512, and so on), which are implemented in the standard Python module `hashlib`. They can take any string as input and always produce an output string that's a hexadecimal representation of the output number of the function with a fixed number of digits:

```
>>> print hashlib.sha1('hello world').hexdigest()
2aae6c35c94fcfb415dbe95f408b9ce91ee846ed
```

Let's look at a simple implementation of a blockchain in Python. First, we define a function that we call *bhash* that, given the timestamp and details (a string or other serializable object) of a new transaction along with the hash of the previous transaction, computes a new hash using the SHA1 algorithm:

```
import hashlib, json, time

def bhash (timestamp, details, prev_hash):
    token = json.dumps([timestamp, details, prev_hash])
    return hashlib.sha1(details).hexdigest()
```

Notice that we used the json serializer to combine the elements together into a hashable string that we then pass to the hash SHA1 hash function. Our choice of serializing in json is an implementation detail and not the only way to achieve the goal.

Next we create a `Blockchain` class to encapsulate a list of blocks:

```
class Blockchain(object):
 def __init__(self, details='new-chain'):
     self.blocks = [(time.time(), details, '')]
 def record(self, details, timestamp = None):
     timestamp = timestamp or time.time()
     prev_hash = self.blocks[-1][2]
     new_hash = bhash(timestamp, details, prev_hash)
     self.blocks.append((timestamp, details, new_hash))
```

The class has a constructor, "`init`", which creates a list of blocks and stores the first block in the list. This first block contains an initial timestamp and details but no hash. In the case of a bitcoin, this would store information about the discovery of a new unit and its owner.

The class also has a second method, "`record`", that, given the details of a new transaction and an optional timestamp (otherwise automatically computed), stores them in a new block. This is done by retrieving the hash of the previous block from `self.blocks[-1][2]`, calling the `bhash` function, and appending the triplet (`timestamp`, `details`, `new_hash`) to the list of blocks. Notice that `self.blocks[i][j]` represents a cell in the blockchain table where `i` is the row number starting from 0, and `j` is the column number also starting from 0.

We use our `Blockchain` class by creating an instance of it, which we call "bc", and recording transactions represented as self-descriptive strings:

```
>>> bc = Blockchain('A found $1')
>>> bc.record('A gives $1 to B')
>>> bc.record('B gives $1 to C')
>>> bc.record('C gives $1 to D')
```

Then we can print the blocks in the blockchain:

```
>>> print bc.blocks
[(1495941516.704196, 'A found $1', ''),
 (1495941516.704201, 'A gives $1 to B', 'a75a9227f...'),
 (1495941516.704277, 'B gives $1 to C', 'ca911be27...'),
 (1495941516.704290, 'C gived $1 to D', 'cb462885e...')]
```

The last hash is `'cb462885e...'`. For this technology to work, we must make sure we broadcast the last hash and that there a few copies of the full chain stored by different parties. The parties in this context are the computing nodes in the peer-to-peer network in charge of recording and storing the transactions. This is a network problem and beyond this article's scope.

It's also important that every party can verify the chain's integrity. This can easily be done by using the function below:

```
def verify(blockchain):
    prev = blockchain.blocks[0]
    for block in blockchain.blocks[1:]:
        new_hash = bhash(block[0], block[1], prev[2])
        if block[2] != new_hash: return False
        prev = block
 return True
```

In the code, above we loop over all the blocks starting from the second one, recompute each hash, and then compare it with the stored one in `block[2]`, the third column. If the code finds any hash that doesn't match, it returns `False`, or else it returns `True`. We can call this code on our blockchain with

```
>>> print verify(bc)
True
```

From a technology viewpoint, there's a lot more than this to the bitcoin network. There are algorithms for data distribution, for syncing nodes, for efficient storage and querying, for conflict resolutions, and so on, yet the blockchain technology is at the heart of it.

Cryptocurrencies and Beyond

It's important to observe that different cryptocurrencies run on different platforms and make different storage and hashing choices. In addition, for the same type of cryptocurrency, for example, bitcoin, there are different implementations of the algorithm, even though they're all compatible and can communicate with each other. Moreover, for each unit of coin, there's one set of blocks (replicated in multiple locations).

Its use for cryptocurrencies is the first and best-known application of the blockchain, but it isn't the only one, and probably not the most important. Many companies provide proprietary implementations of the blockchain technology and sell their solutions to the financial industry, which uses them to record various types of transactions. These proprietary solutions are integrated into the authentication infrastructure of financial institutions and allow different agents to record transactions in a distributed fashion, thereby allowing different institutions (or parts of the same institution) to transact without reciprocal trust.

Because a transaction is basically a string, it can contain arbitrary information. It should be evident to the reader at this point that this technology can be used for any kind of notarization, and not necessarily involving money. For example,

Chicago's Cook County has been experimenting with using the bitcoin network to record house titles (https://bitcoinmagazine.com/articles/chicago-s-cook-county-to-test-bitcoin-blockchain-based-public-records-1475768860). Similarly, someone could store an idea for a patent in the blockchain to later prove a first-to-invent claim. You could also store a promise to do something at a later time, with the promise stored in the form of code that would execute the promise in an automated manner. This is what's called a smart contract; for example, let's say we have this promise: "Alice promises to pay Bob $1 if on 1 January 2028 it rains in Chicago." As long as the promise is in the blockchain, and an API can check whether the conditions are met, the system can automatically execute the transaction should the condition be fulfilled.

The bitcoin network was the first, but new ones are emerging all the time to trade and specifically handle smart contracts, "applications that run exactly as programmed without any possibility of downtime, censorship, fraud, or third-party interference" (thereum.com).

On one hand, the idea of trading cryptocurrencies might be nothing more than stamp collecting, but the other, the underlying technology has only started to revolutionize contracts and human interactions. It will displace many white collar jobs the same way robots have displaced blue collar ones. It will also create new jobs that we can't even imagine today. Only time will tell if cryptocurrencies can soar and prosper because of the increasing trust people put into blockchain technology. ■

Massimo Di Pierro is a professor in the School of Computing at DePaul University and co-director of the MS program in computational finance. Contact him at massimo.dipierro@depaul.edu.