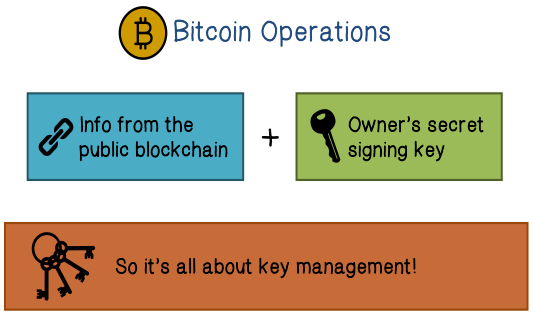
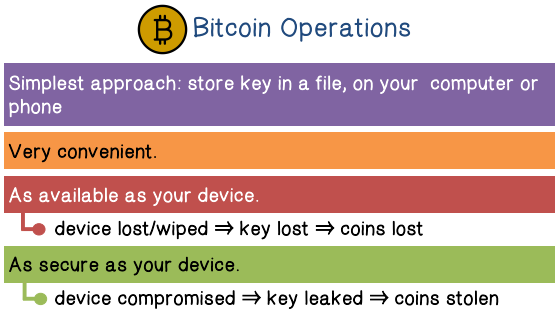
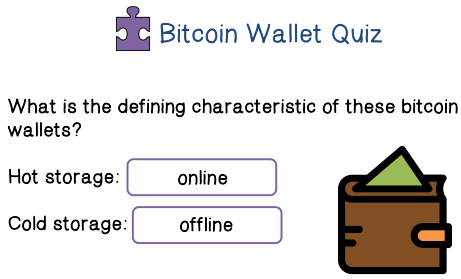
Now that we know the basics of Bitcoin, let us move on to how Bitcoin operates. Since this is network security course, we are going to focus on Bitcoin's vulnerabilities and how to hopefully mitigate them. Before I start, let me acknowledge that the initial slides were created by Joseph Bonneau, Ed Felten, Arvind Narayanan, and Andrew Miller.



Recall that in bitcoin, a valid transaction has information stored in the blockchain, but also the transaction information has to also be signed by the owner's secret signing key, or that's the private key of the public key pair. Recall that in bitcoin, a transaction information is stored in the blockchain and it also contains a signature signed by the owner's private key. The owners need to keep their private key a secret. So, this is about key management. That is, how do you keep your private key secret and secure?



Of course, you can store your public key in a file on your computer or on your phone. This is a very convenient approach. On the other hand, if your key storage is lost, that means your private key is lost, which means all your bitcoins are lost. This is because without the private key, you cannot prove that you are the owner of the matching public key. Remember, all your coins are associated with that public key. That is, your public key is your ID or address in bitcoin. But, you have to use the private key to prove that you are the owner of the public key or the ID. Of course, your key storage can be compromised, which means your key can be stolen, which means all the bitcoins can also be stolen because now the attacker can claim that he is the owner of that private key or the ID. Therefore, he has all your bitcoins. For example, with the stolen private key, now the attacker can claim that he is the owner of the ID that the bitcoins are associated with. He can then pay those bitcoins to another address that he owns. That way, he can steal all the bitcoins.



Now, let us do a quiz on bitcoin wallet. What is the defining characteristic of these bitcoin wallets?

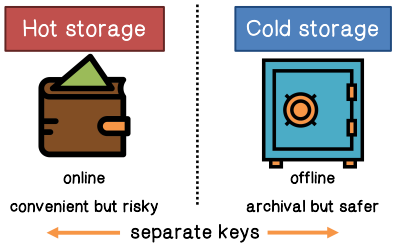
The so-called hot storage is online, and the so-called cold storage is offline.

Bitcoin Wallet Quiz

	Hot Storage	Cold Storage
Location	Online	Offline
Convenient?	Yes	No
Security?	Risky	Archival but safe

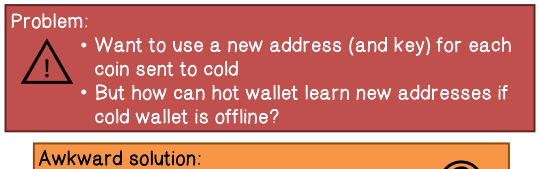
Hot storage is online, for example, on your computer or on your phone. It is very convenient, but it is not very secure. On the other hand, cold storage is offline, not convenient, but it is safe. The reason is obvious, if the storage is offline, then a cyber attacker cannot get to it easily.

Bitcoin Wallet Quiz



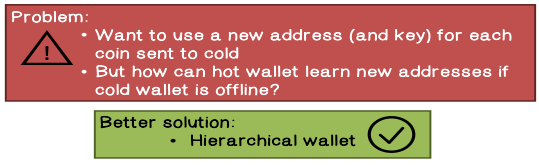
In general, we can keep a top secret in a cold storage and put the most frequent used keys in hot storage. Here is another way to look at the difference between hot storage and cold storage. We can put a top secret or the less frequently used master secret in a cold storage. We can put the most frequently used keys in hot storage. As we just discussed, storing your keys of Bitcoins on computers is storing them in a hot storage. It is online is convenient but somewhat risky. On the other hand, you can put your keys of Bitcoins offline. This is less convenient, but much safer. You can have the best of both worlds by having both hot storage and cold storage. For example, you can move coins back and forth, but then you need to use separate keys and each side needs to know each other's keys or the addresses. You need to use separate keys because otherwise, the coins in cold storage will be vulnerable if the hot storage is compromised. You need to use separate keys because otherwise, the coins in a cold storage will be vulnerable if the hot storage is compromised. You will want to move the coins back and forth between the hot side and the cold side. So, each side will need to note the others addresses or public keys.

Hierarchical Wallet

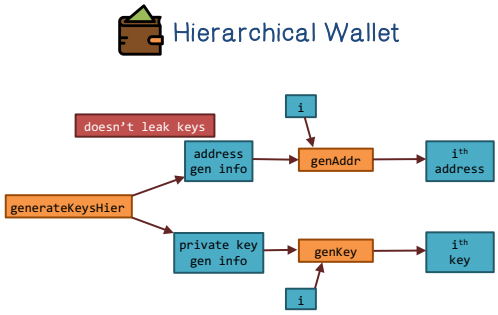


Suppose for privacy or other reasons, we won't to be able to receive each bitcoin at a separate address. So, whenever we transfer coin from the hot side to the cold side, we would like to use a fresh cold edges for that purpose. But, since the cold side is offline, how do the hot side find out those addresses? The blunt solution is for the cold side to generate a big batch of addresses and send those over to the hot side. The drawback is that the hot side need to reconnect with the cold side periodically in order to transfer more addresses.

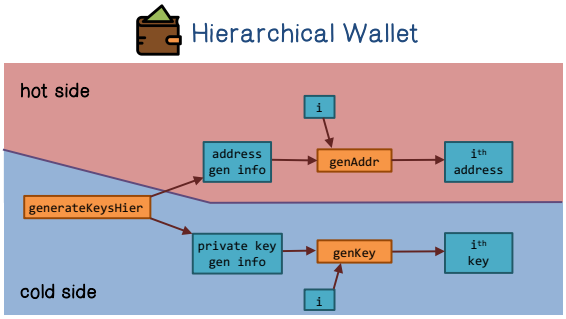
Hierarchical Wallet



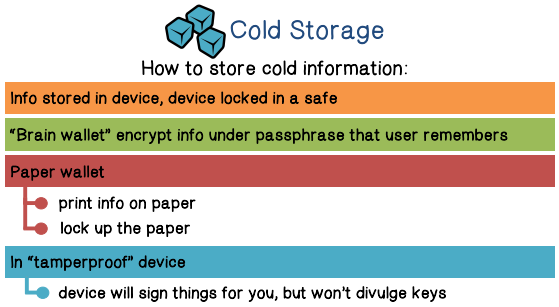
A better solution is to use the so-called hierarchical wallet.



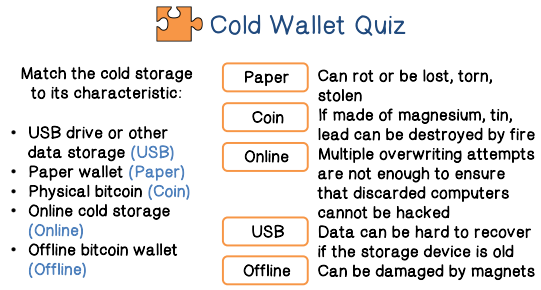
To review, previously we discussed key generation and digital signatures, and we looked at a function called generateKeys. This function generates a public key which acts as an address and a secret or private key. In a hierarchical wallet, key generation works in different way. Instead of generating a single address, we generate what we called address generation information. Rather than private key, we generate what we call private key generation information. Given the address generation information, we can generate a sequence of addresses. Likewise, we can generate a sequence of public keys using the private key generation information. The cryptographic magic that makes this useful is that for every i, the ith address and the ith secret key match up. That is, the ith secret or private key controls and can be used to spend bitcoins from the ith address. That is, the ith private key is paired up with ith address. In other words, we have a sequence of regular key pairs. The other important cryptographic property here is security, that is, the address generation information here does not leak any information about the private keys. On the other hand, not all digital signature schemes can be modified to support hierarchical key generation. The good news is that the digital signature scheme used by bitcoin, elliptic curve, does support hierarchical key generation. As a result, the cold side can generate as many keys as it wants and the hot side can generate the corresponding addresses.



To summarize, the cold side create and saves private key generation information and address generation information. It has a one-time transfer of the address generation information to the hot side. The hot side generates a new address sequentially every time it wants to send a coin to the cold side. When a cold side reconnects, it generates addresses sequentially and checks the blockchain for transfers to these addresses. It can also generate private keys sequentially if it wants to send some coins back to the hot side.

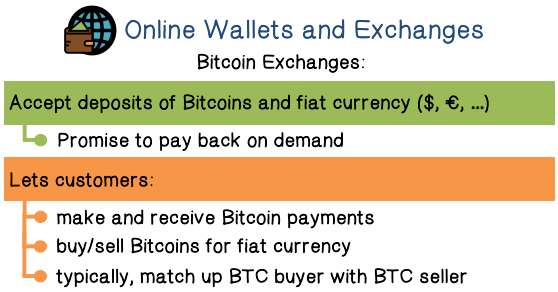


Now, the remaining question is, how do we keep the cold storage safe? There are multiple options. For example, a user can use his pass phrase to encrypt information. We can even print information on a piece of paper, or we can use tamperproof device then use hardware to protect the keys.

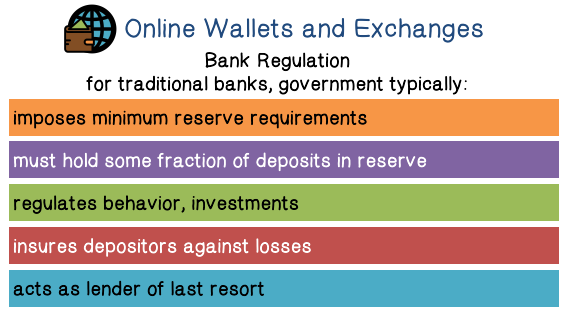


Now, let's do a quiz. Match the cold storage to its characteristic.

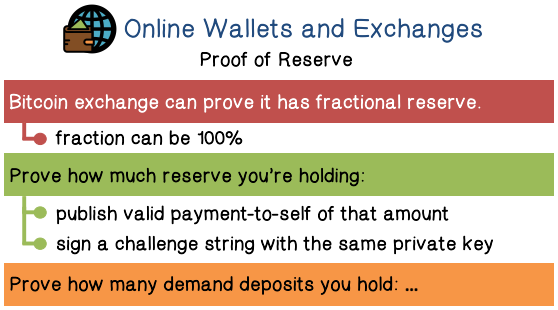
First, it can rot or lost, or be torn, or stolen. That's paper storage. Second. If make of magnesium, tin, lead, it can be destroyed by fire. That's coin. Third. Multiple overwriting attempts are not enough to ensure that discarded computers cannot be hacked. That's online cold storage. Fourth. Data can be hard to recover if the storage device is old. That's USB. Fifth. Can be damaged by magnets. That's offline Bitcoin wallet.



Now, let's discuss bitcoin exchange. Bitcoin exchange accepts deposits of Bitcoins and fiat currency. For example, dollars, pounds, euros, and so on. These exchanges let customers make and receive bitcoin payments, and buy and sell bitcoins using fiat currency. You would typically match a bitcoin buyer with a bitcoin seller. Of course, it promises to give the money back to the customer, therefore, bitcoin exchanges are similar to banks.

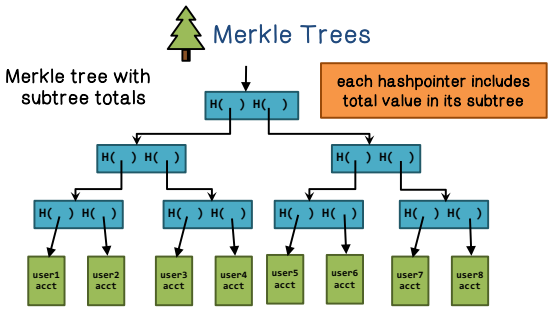


As a result, bitcoin exchanges need to meet a number of minimal requirements or expectations. For example, bitcoin exchanges should meet the minimum reserve requirements, that means they must hold some fraction of deposits in reserve.

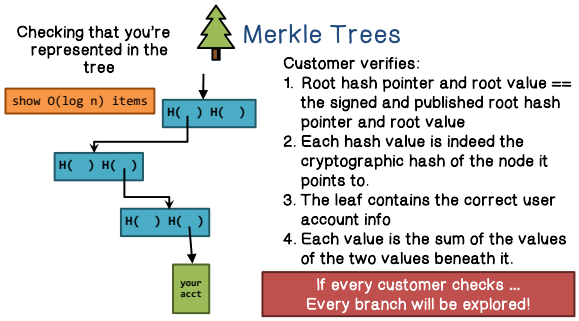


A Bitcoin exchange needs to provide a proof of reserve to give customers some comfort about the money that they have deposited. The goal is for the exchange to prove that he has a fraction or reserve, for example, from 35 percent to even 100 percent of the deposits that people have made. We can break the Proof of Reserve problem into two pieces. The first is to prove how much reserve you are holding. The exchange can simply publish a valid payment-to-self transaction of the claimed

reserve amount. That is, if he claims to have 100,000 bitcoins, it creates a transaction in which he pays 100,000 bitcoins to herself and show that the transaction is valid. Strictly speaking, this is not a proof that the exchange really owns this amount, but only that whoever does own this amount is willing to support the exchange. The second piece of the Proof of Reserve problem is to prove that how many demand deposits you hold. We will discuss this next.

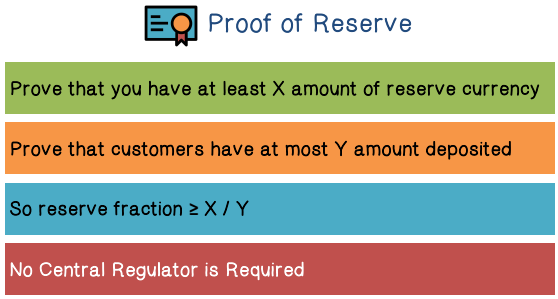


The second piece of proval reserve is to prove how many demand deposits you hold, and here is a scheme. Recall that a Merkle tree is a binary tree that is built with hash pointers, so that each of the pointers not only sets where we can get a piece of information, but also what a cryptographic hash of that information is. A bitcoin exchange executes a prove of how many demand deposits it holds by constructing Merkle tree. In this Merkle tree, each hash pointer also includes an attribute. This attribute is a number that represents the total value in bitcoins of all of the deposits that are in a subtree underneath this hash pointer. For this to be true, the attribute value of each hash pointer has to be the sum of the values of the two hash pointers beneath it. A bitcoin exchange constructs this tree and size the root pointer along with the route attribute value and publishes it. The root value is, of course. The total liabilities. That is, by constructing this tree and publishing its root value, the exchange is making the claim that all users are represented in the leaves of the tree and that the deposit values are represented correctly. Furthermore, the deposited values are propagated correctly up the tree, so that the root value is the sum of all the users' deposit amounts.



Now, each customer can go to the exchange and ask for proof of correct inclusion of his deposit. The exchange must then show the customer the puzzle tree from the user's leaf up to the root. The customers then verify that first. The root hash pointer and root value are the same as to what exchange has signed and published. Second, the hashpointers are consistent all the way down. That is, each hash value is indeed the cryptographic hash of the node it points to. Third, the leaves contain correct use account information. For example, user ID and the deposit amount. Fourth, each value is the sum of the values of the two values beneath it. The good news here is that if every customer does this, then every branch of the tree will get explored, and someone will verify that for every hash pointer, its associative value equals the sum of the values of its two children. It's very important that the exchange cannot present different values in any part of the tree to different customers. That's because doing so would either imply that the ability to find a hash collision or presenting different root values to different customers which we assume is not possible. The customer then verifies that, first, the root hash pointer and root value are the same as what the exchange has signed and

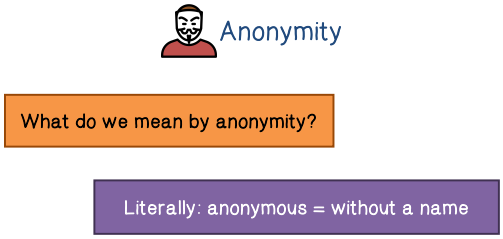
published. Second, the hash pointers are consistent all the way down. That is, each hash value is indeed the cryptographic hash of the node it points to. Third, the leaf contains correct user account information. For example, user ID and a deposit amount. Fourth, each value is the sum of the values of the two values beneath it. The good news is that if every customer does this, then every branch of the tree will get explored, and somehow we verified that for every hash pointer it's associative value equals the sum of the values of its two children. It is very important to note that the change cannot present different values in any part of a tree to different customers because that will imply that either the exchange can find a hash collision, or it can present different root values to different customers, which we think is not possible.



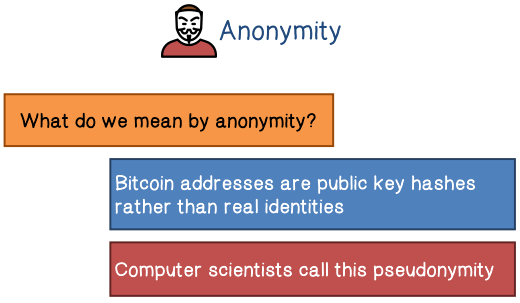
Let's recap. To provide a proof of reserve, first, the exchange proves that they have at least X amount of reserve currency by doing a self transaction of X amount. Then they prove that the customers have at most amount of Y deposited. This shows that their reserve fraction is at least X divided by Y. Notice that a bitcoin exchange can provide this proof, that is independently verifiable by anybody, and therefore, no central regulator is required.

Now, let's discuss anonymity in Bitcoin. Some, including WikiLeaks, claims that BitCoins provide anonymity.

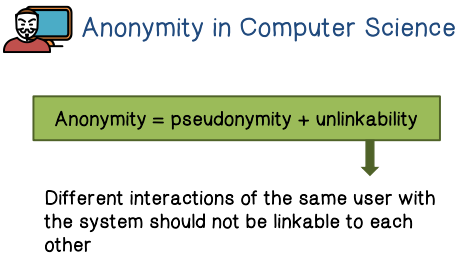
Others, believe that Bitcoin does not provide anonymity.



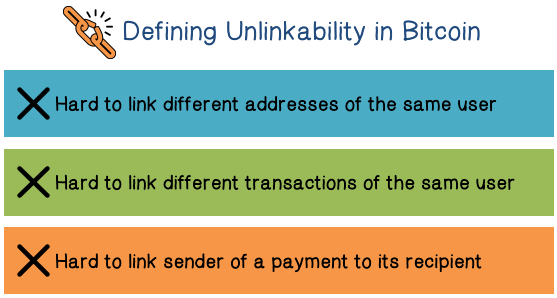
So, what is anonymity? Literally, anonymous means without a name.



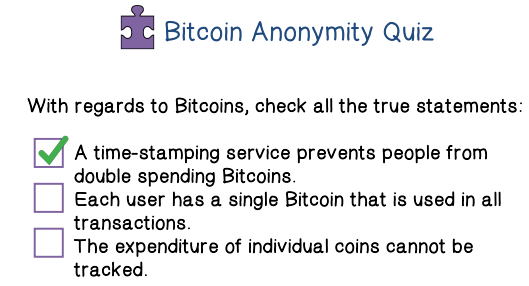
Bitcoin, an ID is an address, which is the hash of a public key. In computer science, we call this pseudonymity.



In computer science, an anonymity means pseudonymity plus unlinkability. We know that Bitcoins provides pseudonymity. Now, the question is does Bitcoin provide unlinkability. That is, can we be sure that in Bitcoin, different transactions of the same users can be linked together.



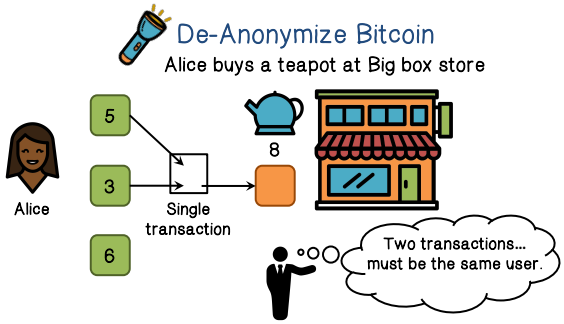
More precisely, unlinkability in Bitcoin means that, it is also hard to link different transactions of the same user. Furthermore, it should be hard to link the sender of a payment to its recipient.



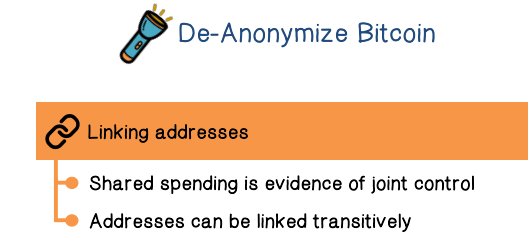
Let's do a quiz on bitcoin anonymity. Check all the true statement.

First, a timestamping service prevents people from double-spending Bitcoins, this is true. Second, each user has a single Bitcoin that is used in all transactions, this is obviously false. Third, the expenditure of individual coins cannot be tracked, this is false because individual coins can be tracked by their chain of digital signatures.

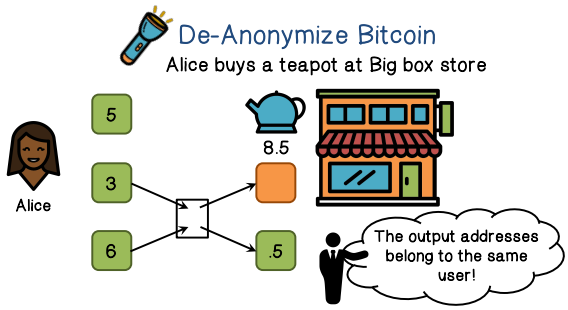
We know that in Bitcoin, it is trivial to create a new address. That is, all you need to do is to create a new public key public key pair and a hash of the public key becomes a new address. That is, a single user can own many addresses or many IDs. In fact, the best practice is to use a fresh new address for the recipient in any new transaction. So, the question is, can the addresses and the transactions still be linked?



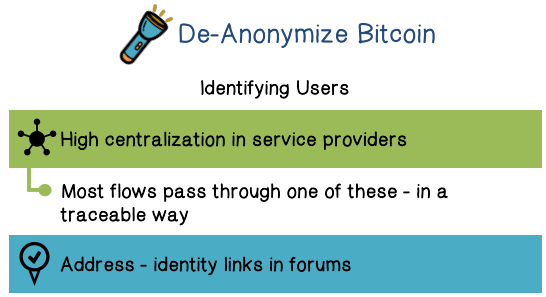
Let's consider an example. Suppose Alice wants to buy a teapot from a store and a teapot cost eight Bitcoins. Let's further assume that Alice has some Bitcoins in her different addresses. In order to pay for this teapot, Alice has to create a single transaction that includes input from two addresses. But in doing so, Alice has revealed that these two addresses belong to the same user. In other words, two previous transactions that use these two addresses as recipients must be to the same user.



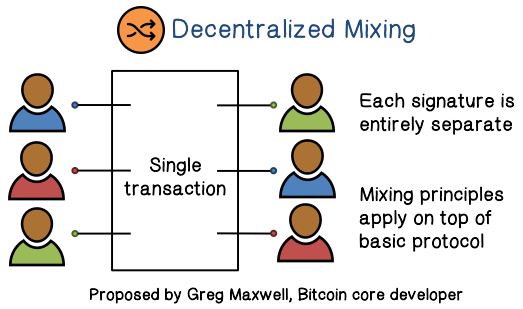
This example that if you're sure to spending for multiple addresses, we're linking these addresses together. This example shows that if multiple addresses are using the same spending transaction, that means that these addresses are under joint control. In other words, we can link these addresses together. Further, address linkability is tentative. Furthermore, we can propagate such linkability.



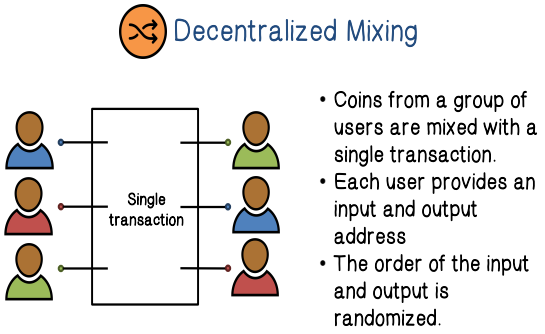
Let's consider another example. Suppose now, the price of the teapot has gone up from eight Bitcoins to 8.5 bitcoins. So, Alice will create a transaction where she will transfer Bitcoins from two of her addresses to pay for the teapot and also send the change to her own address. Therefore, Alice will create transaction where she transfers Bitcoins from two of her addresses but also send a change to one of her addresses as well. So, Alice will create a transaction where she would transfer Bitcoins from two of her addresses to pay for a teapot but also send a change to another address of hers. Now, let's consider the transaction from the viewpoint of an adversary. He can deduce that these two input addresses belong to the same user. The attacker might also suspect that one of these output addresses also belongs to the same user. If the attacker knows that a teapot cost more than 0.5 Bitcoin, then he would know that this address also belongs to the same user. Therefore, in this example, the attacker can know that these three addresses belong to the same user.



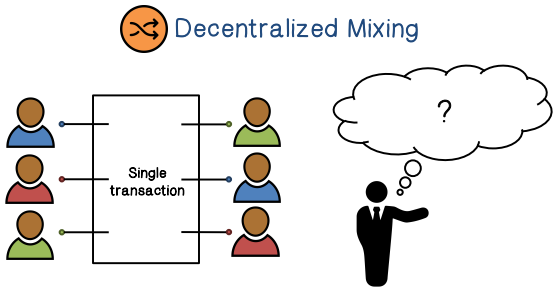
We can only link or cast the addresses that belong to the same user. We can also see if any of these addresses reveals the real identity of a real user. For example, an address may be used in transaction through a provider that is legally required to know the true user behind the address or a user may have posted an address in a public forum. For example, because he's making a donation.



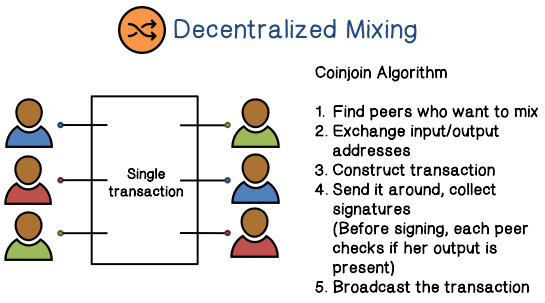
One mechanism to make transaction link analysis less effective, and let's protect unlinkability, is to use mixing. The main proposal for decentralized mixing. Since bitcoin is a decentralized system is called coinjoin. In this protocol, different users joining, create a single bitcoin transaction that combines all the inputs. Furthermore, in a transaction that has multiple inputs coming from different addresses, the signatures corresponding to each input are separate from and independent of each other.



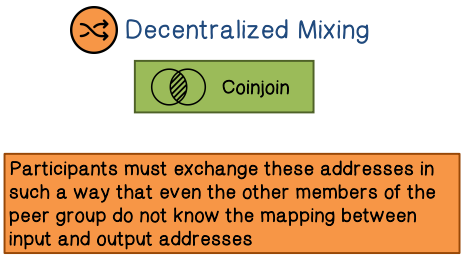
This allows a group of users to mix their coins with a single transaction. Each user supplies an input address and an output address and form a transaction with these addresses. The order of input and output addresses is randomized, so that an outsider will be unable to determine the mapping between inputs and outputs. All the participants check that it's outgoing address has been included in the transaction and that it receives the same amount of bitcoin that they are inputting. Once they have confirmed this, they sign the transaction.



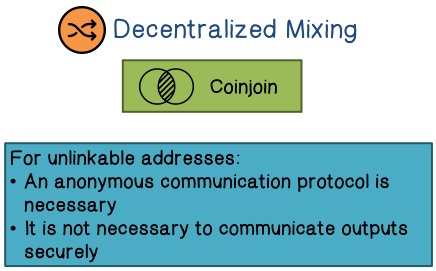
Somebody, say an attacker looking at this transaction on the blockchain, even if they know that it is a coinjoin transaction, would be unable to determine the mapping between the inputs and outputs. From an outsider's perspective, the coins have been mixed and this is the essence of coinjoin.



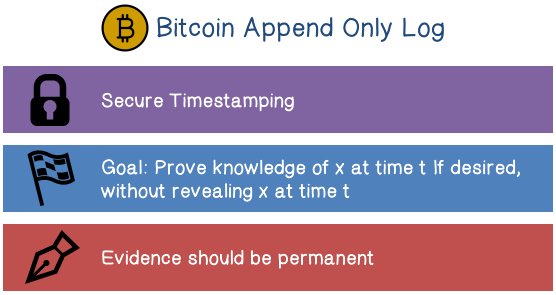
Here's the procedure to create coinjoin. First, a participant needs to find peers who want to mix. Then, they exchange the input and output addresses to be included in the transaction. Then, they construct the transaction by mixing the orders. Then, the transaction is sent around and each peer can check that his or her input and output have been included. Then, the signatures are collected in a transaction. Finally, this coinjoin transaction is broadcast to the bitcoin system.



It is important for the participants to exchange these input and output addresses in such a way that even the other members of the peer group do not know the mapping between input and output addresses.

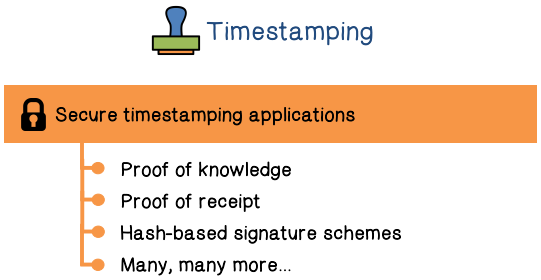


To exchange these addresses in unlinkable way, we need an anonymous communication protocol. For example, we can use Tor or special purpose protocol. For example, the peers can use Tor to exchange the input addresses. Therefore, no one knows what input address each other is using. Once that is accomplished, it is not necessary to communicate the output addresses in a secure way. Once that is accomplished, there's no way to link an output address with an input address.

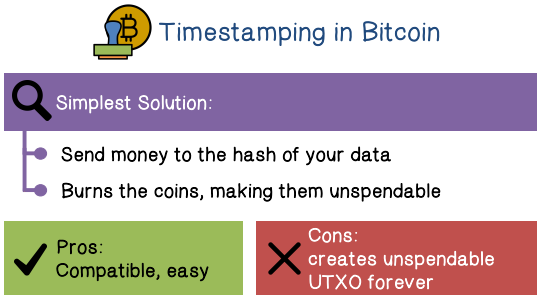


The Bitcoin system can be used for other applications. For example, secure timestamping. For example, we can prove knowledge of x or we know the value of x at a time t. But we can do so without revealing the actual value or knowledge of x at time t. Of course, if we choose to, we can review x at some later time.

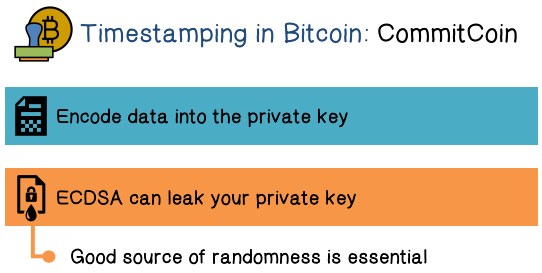
Recall that hash function is a one-way function. That is, if you publish the hash value of x, you're announcing a commitment to x. Recall that a cryptographic hash function is a one-way function and also loses in to collision. Therefore, if we publish the hash value of x, we essentially announcing the commitment to x. In other words, everyone will know that we actually know the true value of x by publishing the hash value of x, and we don't have to reveal x to anyone. In other words, we can later prove that we actually knew the value of x because we had published the hash value of x.



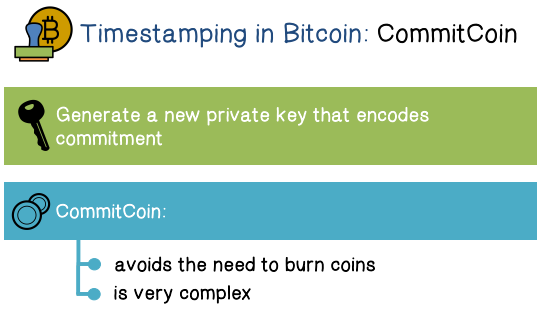
Secure timestamping has many applications. We have discussed proof of knowledge. We can also use it as proof of receipt and so on.



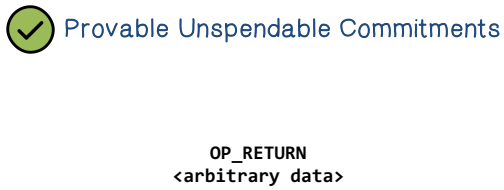
As we have discussed, to make a commitment, we need to publish the hash value of the data. The simplest solution in bitcoin system is that instead of sending money to the hash of a public key, just send it to the hash of your data, and then by announcing the transaction, you're announcing the hash of your data. The caveat here is that the coin you sent can be lost forever because you don't know who actually happens to own the address that corresponds to the hash of your data. This approach is indeed very simple, but we have to burn coins. More importantly, the bitcoin miners don't know that the coin you send to this address is lost forever. So, they must check it forever just like any other legitimate bitcoins.



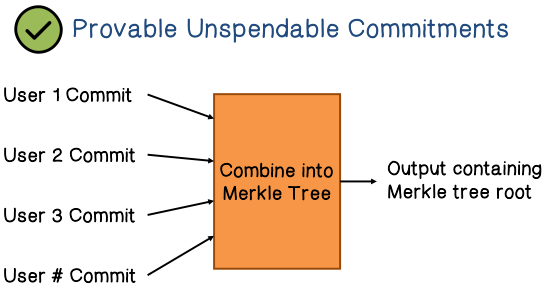
A more sophisticated approach is called CommitCoin. It allows you to encode your data into the private key. Recall that elliptic curve public key is used in bitcoin. Just like other public key systems, we need good randomness, otherwise the private key can be leaked. A property of elliptic curve is that if you use bad randomness in making a signature, it will leak the private key.



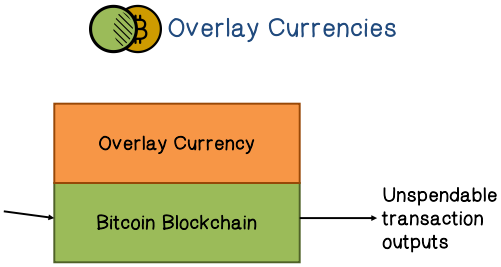
CommitCoin exploits this property. CommitCoin generates a new private key that encodes a commitment and derive its corresponding public key. We can then send a tiny transaction to that address that corresponds to the public key and then the address also sends back another transaction and we use the same randomness in signing both of these transactions. In CmomitCoin, we generate a new public key that encodes our commitment, and we derive its corresponding public key. We then send a tiny transaction to the address that corresponds to the public key and we then send it back two chunks of bitcoins. When doing so, we'll use the same randomness to sign both chunks in the transaction and when sending it back, we'll use the same randomness both times for signing the transaction. This allows anyone looking at the blockchain to compute the public key using these two signatures and the private key contains the commitment. CommitCoin avoids the need to burn coins and the miners don't have to keep track of unspendable output forever. However, it is quite complex.



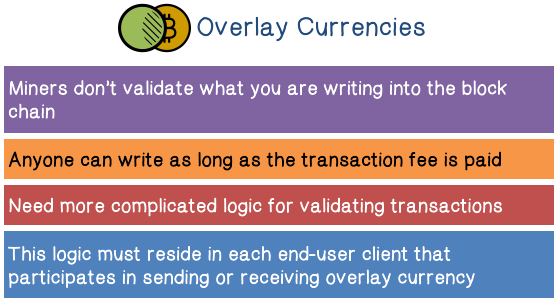
As of 2014, the preferred way to do bitcoin timestamping is with the OP_RETURN transaction. The OP_RETURN instruction returns immediately with an error, so that this script can never be run successfully, and the data that is encoded in the transaction is ignored. This can be used to encode arbitrary data. As of 2015, OP_RETURN allows 80 bytes of data to be pushed, which is more than enough for a hash-function output. For example, shot to 56, the output length is 32 bytes.



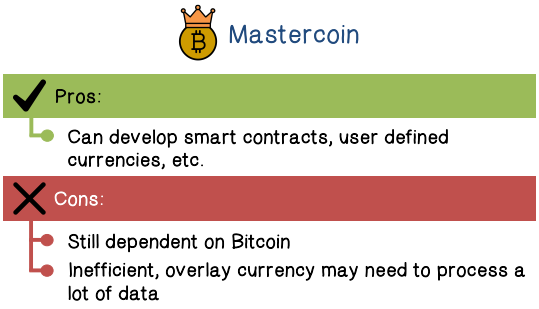
As of late 2014, there are already several websites that help with this. They collect a bunch of commitments from different users and combine them into a large Merkle tree, then they publish one unspendable output containing the Merkle tree root. This acts like a commitment for all the data that users wanted to timestamp that day.



Since we can write whatever data we want into Bitcoin, we can also view an entirely new currency system on top of Bitcoin, without needing to develop a new consensus mechanism. That is, we can simply use Bitcoin as it exists today as an append-only log and write all the data that we need for our new currency system directly into the Bitcoin blockchain. We call this approach an overlay currency. That is, Bitcoin serves as the underlying substrate, and the data of the overlay currency is written into the Bitcoin blockchain using unspendable transaction outputs.



Of course, Bitcoin miners will not actually validate what you're writing into the blockchain because they don't know the new currency system. Also, anyone can write anything as long as they're willing to pay the transaction fees. Since this is a different new currency system, you must develop your own logic for validating transactions.



An example of overlay currency is Mastercoin. In an overlay currency system, such as Mastercoin, there's no need to develop a new consensus algorithm and, therefore, developers can instead focus on developing interesting features, such as smart contracts. On the other hand, such an approach can also be inefficient because those on the overlay currency may need to process a lot of data. This is because bitcoin nodes don't filter these transactions for you.