Network Security

Professor Gaurav S. Kasbekar

Department of Electrical Engineering

Indian Institute of Technology, Bombay

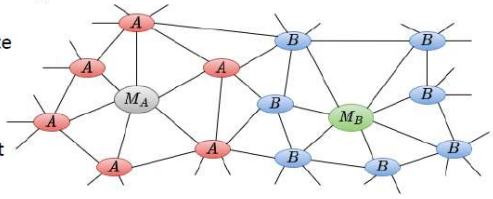
Week - 11

Lecture - 64

The Bitcoin Cryptocurrency: Part 4

Hello, in this lecture, we will continue our discussion of the Bitcoin cryptocurrency. We discussed blockchain forks. We have discussed that different miners try to mine blocks in parallel. What happens if two miners find valid blocks at around the same time and broadcast the blocks? For concreteness, assume that initially all the network nodes have the same copy of the blockchain, which ends in a block at height N-1. This is the blockchain. This is block 0, and so on and so forth.

- For concreteness, assume that:
  - initially, all the network nodes have the same copy of the blockchain, which ends in a block at height $N-1$
  - two valid blocks, A and B, both at height N, are found by two different miners, M_A and M_B, respectively, and are broadcasted before one of M_A and M_B received the other miner's block
- The other nodes either receive block A first or block B first
- Each node accepts the first block at height N that it receives and rejects the second block, e.g.,
  - if a miner receives block A first, it appends it to its local copy of the blockchain and starts mining for a block at height $N+1$ with block A as the previous block
  - if the miner later receives block B, it will reject it
- Eventually each node would have received either block A or B and appended it to its local copy of blockchain at height N
- Such a situation called a *blockchain fork* since the state of the network as seen by network as a whole consists of two branches both originating from same parent block at height $N-1$



Block N-2 and block N-1. So, all the nodes agree on this copy of the blockchain; that is, it's a linear list of blocks up to the block at height N-1. Now, two valid blocks, say A and B, both at height N, which is the next position of the block. So, these two valid blocks, A and B, at height N are found by two different miners. Let's call them M_A and M_B, respectively.

They are broadcast before one of M_A or M_B receives the other miner's block. So, miner M_A finds block A and broadcasts it to the other miners. And almost simultaneously, miner M_B

mines block B and broadcasts it on the network. The other nodes either receive block A first or block B first. So, typically, the nodes that are close to miner M_A will receive block A first, and the nodes that are close to miner M_B will receive block B first.

Each node accepts the first block at height N out of A or B that it receives and rejects the second block. For example, if a miner receives block A first, then it appends it to its local copy of the blockchain and starts mining for a block at height N+1 with block A as its previous block. So, a miner which receives block A first, it will append block A at height N, and then it will start mining for the next block at height N+1. If the miner later receives block B, then it will reject it because it has already got a block at height N. So, block B is another block at height N. So, it will reject the block.

So, it accepts the first block at a particular height that it receives. Eventually, each node would have received either block A or block B and appended it to its local copy of the blockchain at height N. So, this particular node in this example happened to receive block A first and appended it at height N. But it can happen that almost simultaneously another miner received block B before block A and appended it to its copy of the blockchain. So, this results in a situation like this, where the blockchain is the same up to height N-1 but subsequently there is a fork. Some miners have appended block A to the blockchain, and some miners have appended block B to the blockchain at the same height N. So, this shows another picture: this is miner M_A, which found block A, and this is miner M_B, which found block B. The neighbors of miner A receive block A first before block B and append block A to the local copy of their blockchain at height N, and their blockchain looks like this—the upper part of the blockchain.

This is miner M_B and its neighbors receive block B first and they append block B to their local copy of the blockchain. So, their blockchain looks like this. The lower branch is there in their blockchain. The situation is called a blockchain fork. So, that fork is illustrated here.

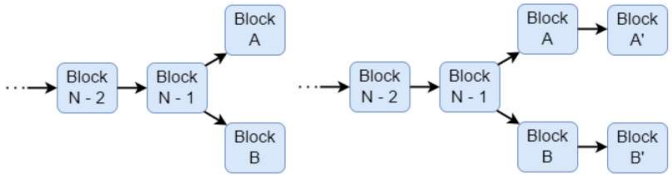
There are two branches from this point. Since the state of the network, as seen by the network as a whole, consists of two branches originating from the same parent block at height N-1. So, we can see that there is a common parent block at height N-1, and there are two branches emanating from this parent block at height N-1. This is known as a blockchain fork. So, a blockchain fork is shown in the figure on the left.

It's the same figure that we drew on the previous slide. All the nodes agree on the blockchain up to height N-1, but then there is a fork. Some nodes add block A at height N,

and some nodes add block B at height N. So, this figure on the left shows a blockchain fork. Now, there are two branches: one ending at block A and another ending at block B. Both branches will have some proportion of the miners in the Bitcoin network working to extend them. So, the miners which appended block A to their local copy of the blockchain, they will work to extend the blockchain starting from block A and those miners which received block B and appended it to the local copy of the blockchain, they will work to extend the blockchain starting from block B. It is possible that valid blocks are again found around the same time on both the branches and broadcast on the network.

So, one of the miners who was working on extending this branch found a block, and almost at the same time, a miner who was working to extend this branch found a valid block as well. So that results in the situation, that is, shown in this figure on the right, where the branches are now extended. Again, the lengths of the branches are the same. So, in this case, blocks A' and B', which is this block and this block, they were found by miners trying to extend the branches containing blocks A and B respectively. Now, how is a blockchain fork resolved?

- This results in situation shown in fig. on right
  - ❑blocks A' and B' were found by miners trying to extend the branches containing blocks A and B, respectively
- How is a blockchain fork resolved?

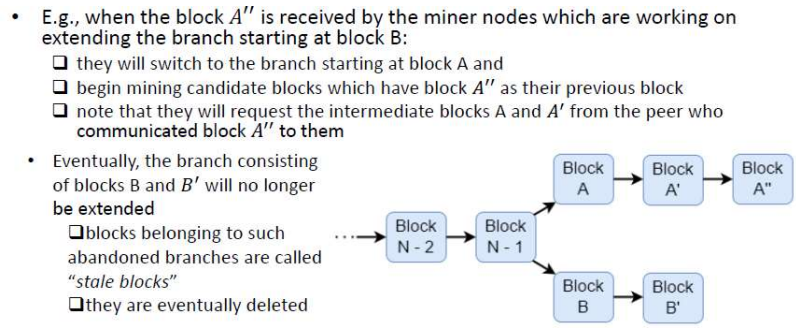


At any time, the blockchain should be a linear list of blocks. It should be a chain of blocks. There should not be any fork in the blockchain. So, how is such a blockchain fork resolved? We'll now discuss how a blockchain fork is resolved.

Recall that there is randomness inherent in the mining process. The time taken to mine a particular block is not constant. It depends on when a particular miner happens to find a block whose block header has a double SHA-256 hash less than the threshold. So, that's a random process. So, the time until a miner successfully mines a valid block, it's a random time.

So, it's unlikely that both branches in the blockchain fork will extend to equal heights indefinitely. Eventually, one branch will become longer than the other. An example of such

a situation is shown in this figure. There are two branches starting from height N, one starting from block A and one starting from block B. And it so happens that a miner who is working to extend the upper branch finds a block A" before any miner who is working to extend the lower branch happens to find a valid block.



So, the upper branch has now become longer than the lower branch. The branch starting from block A has been extended to height N+2 while the branch starting from block B has been extended to height N+1. So, this is the technique used for resolution of blockchain forks. The Bitcoin protocol requires the network nodes to switch to the longest branch that they become aware of. So, in this example, when the block A" is received by the minor nodes which are working on extending the branch starting at block B, they will realize that they are trying to find a valid block at this height, that is, height N+2, but some miner has already sent them a valid block at height N+2.

Then the miners will switch to the branch that starts at block A, because that's a longer branch. And following this rule, that they have to switch to the longest branch that they become aware of. Then the miners will begin mining candidate blocks which have block A" as their previous block. Their local copy of the blockchain will look like this with the upper branch included and they will work on finding a valid block after block A". Note that they will request the intermediate blocks A and A' from the peer who communicated block A" to them.

Eventually, the branch consisting of blocks B and B' will no longer be extended because all the miners in the network will eventually receive this latest block of the upper branch and then they will abandon the lower branch and they will work on extending the upper branch. Blocks belonging to such abandoned branches are called stale blocks. So, these blocks are abandoned and the blockchain is at all the nodes becomes this, with the upper branch appended to the chain. So, this way the fork is resolved. So, such stale blocks like block B and B' in this example, such stale blocks are eventually deleted.

So, in summary, by having all the nodes switch to the longest branch, the Bitcoin protocol ensures that only a single linear list of blocks survives after the resolution of blockchain forks. The network is said to have achieved consensus about which linear list of blocks constitutes the blockchain. So, in the previous example, does this branch constitute the blockchain, or does this one constitute the blockchain? So, that is resolved after the resolution of the blockchain fork in this manner, and the network achieves consensus. Now, the question is: what happens to the transactions in the stale blocks?

So, in this example, we have seen that blocks B and B' are deleted, but in each of these blocks there was a coinbase transaction and there was some regular transactions. So, what happens to these transactions? Let's discuss what happens to these transactions. A transaction is valid only if it belongs to a block which survives after any blockchain forks have been resolved. Recall that the coinbase transaction encodes the transfer of the block reward to the miner who creates the block. The coinbase transactions in the stale blocks become invalid.

So, in particular, the miners who mined the stale blocks, they will lose the block reward which they had gained. What happens to the regular transactions? A regular transaction in a stale block may already be present in one of the blocks that survived after the fork resolution. This is because each node just collects some regular transactions from the mempool and puts them in its block and tries to find a nonce such that the block header has double SHA-256 hash less than the threshold. So, it can happen that, in parallel, several nodes added the same regular transaction to the block they were mining.

Hence, a regular transaction in a stale block may already be present in one of the blocks which survived after the fork resolution. If not, then it is added back to the mempool of transactions which nodes use to construct the new candidate blocks. So, in the previous example, consider the miners, for example, who had created blocks B and B'. When these became stale and when they switched to the upper branch, when they mine the next block, they will see that the transactions, regular transactions that were earlier in blocks B and B', they are not in any of the blocks in the blockchain. So, they will put them back into the candidate block and try to find a valid block.

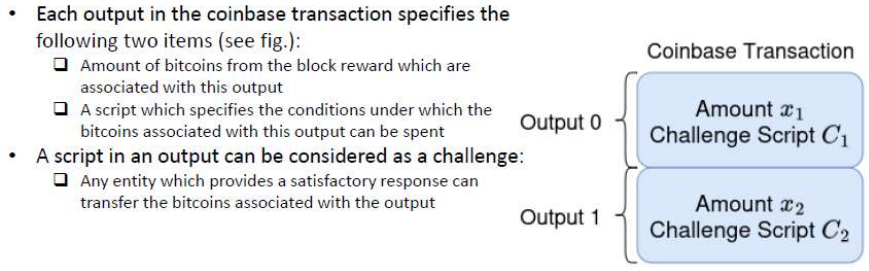
So, these regular transactions are eventually again put in valid blocks and they become part of the blockchain. This concludes our discussion of the resolution of blockchain forks. Now, we discuss transactions in more detail. Recall that each block contains one coinbase

transaction and zero or more regular transactions. A Bitcoin transaction encodes the transfer of bitcoins between different entities.

A destination of the transfer in a transaction is called an output. A single transaction can have several outputs. For example, in the case of the coinbase transaction encodes the transfer of the block reward to the miner who created the block. So, in that case, the block reward may be split among multiple outputs. Each output can serve as a source of bitcoins in a subsequent transaction.

When previous transaction outputs are specified as sources of bitcoins in a transaction then they are called inputs. Each regular transaction has one or more inputs and one or more outputs and a coinbase transaction does not have any inputs but it has one or more outputs. So, a coinbase transaction has no input and has at least one output. The question is why is there no input in a coinbase transaction? The reason is that the bitcoins in the coinbase transaction, they do not come from any previous transaction, but they are from the block reward, which the miner who creates the block gets.

So, there is no input in a coinbase transaction because the source of bitcoins is not a previous transaction output, but it's the block reward, that is, the sum of the block subsidy and the transaction fees from the transactions in the block. Now, each output in the coinbase transaction specifies the following two items. These are shown here. So, this is a coinbase transaction and in this example there are two outputs. Let's call them output 0 and output 1.



Each output has an amount and a challenge script. So, the first item that is there in every output is the amount of bitcoins from the block reward which are associated with this output. So, in this example, an amount of x_1 bitcoins from the block reward correspond to this particular output, output 0, and an amount x_2 from the block reward correspond to this output, output 1. Then the other item that is there in every output is a script which specifies the conditions under which the bitcoins associated with this output can be spent. So, we'll

see that this challenge script has a public key and someone who wants to respond to this challenge correctly must know the corresponding private key.

A script in an output can be considered as a challenge. Any entity which provides a satisfactory response can transfer the bitcoins associated with the output. This script has a public key and only someone who knows the corresponding private key can transfer the bitcoins corresponding to this output to some other node in the bitcoin network. Now, the sum of the amounts in all the outputs of the coinbase transaction should not exceed the block reward because the block reward is distributed among the different outputs. For example, in this figure, suppose the block reward is R, then what is the relation between x_1, x_2, and R?

- Then $x_1 + x_2 \leq R$
- Typically equality holds
- If $x_1 + x_2 < R$, then:
  - the $R - x_1 - x_2$ bitcoins from the block reward become unspendable
- Note that a single output in the coinbase transaction is sufficient for a miner to gain control of the block reward

So, the relation is $x_1 + x_2 \leq R$. Typically, $x_1 + x_2$ will be equal to R. The block reward will be split among the different outputs of the coinbase transaction. What if $x_1 + x_2 < R$? In that case, the $R - x_1 - x_2$ bitcoins from the block reward become unspendable. So, out of these R bitcoins of the block reward, x_1 correspond to this output, they can later be spent by the miner. x_2 correspond to this output.

They can be later spent by the miner. And the remaining $R - x_1 - x_2$ are wasted. They become unspendable. So, for this reason, $x_1 + x_2$ will be typically equal to R, except when there is some error. In that case, $x_1 + x_2$ may be less than R. But in most cases, $x_1 + x_2 = R$. Note that a single output in the coinbase transaction is sufficient for a miner to gain control of the block reward.

So, there could have been just one output whose amount was R, the entire block reward. Then, where were multiple outputs used? Each input in a regular transaction unlocks all the bitcoins associated with the previous transaction output. Hence, multiple outputs give the miner flexibility to distribute the block reward to multiple addresses. As an example, suppose the block reward R is, say, 15 bitcoins.

So, one possibility would have been to just create one output whose amount is 15 bitcoins. But then the miner would have to give the entire 15 bitcoins to only one entity. So, a more

flexible option is to create three different outputs, say 0, 1, 2, and 3, and the corresponding amounts are these: $x_1=5$, $x_2=5$, $x_3=5$. So, in this case, the miner creates a coinbase transaction in which there are three outputs, and the amounts in these three outputs are 5 each. So, in this case, the miner has the flexibility to pay 5 bitcoins to three different parties.

Hence, this is the reason for having multiple outputs in a coinbase transaction. Here's an example. We now study an example of a challenge script and a satisfactory response to the challenge script. Consider a miner who creates a block. The miner wants the block reward to be transferred to addresses that the miner owns.

So, these are public keys of the miner. An address is a public key of the miner, and it is called a "Pay-to-Public-Key (P2PK) address." The ownership of a P2PK address means knowing the corresponding private key. So, the miner has pairs of private keys and the corresponding public keys, and the challenge in each output is the public key, and the miner knows the corresponding private key. The challenge script in an output of the coinbase transaction will contain a P2PK address.

This challenge script will require anyone who wants to spend the bitcoins to provide a response script which contains a digital signature that is created using the private key corresponding to the P2PK address. So, only someone who knows the corresponding private key—the private key corresponding to the public key appearing in the challenge script—can spend this output and transfer these bitcoins to another party. Note that this digital signature can be verified using the public key, that is, the P2PK address already present in the challenge script. So, anyone can see the public key and verify the digital signature. So, we discussed coinbase transactions. Now, we discuss regular transactions.

To spend the bitcoins earned in a coinbase transaction, the miner needs to create a regular transaction. So, in the previous example, the miner had earned 15 bitcoins and created a coinbase transaction with 3 outputs, each corresponding to the amount 5 bitcoins. If the miner wants to spend the 5 bitcoins from one of these outputs, then the miner will have to create a regular transaction. Regular transactions have at least one input and at least one output. Inputs are the sources of the bitcoins in the regular transactions and the outputs are paid to the parties to which the transaction creator wants to transfer the bitcoins to.

So, each input specifies the following three items. One is the transaction ID of a previous transaction on the blockchain. So, the transaction ID of a transaction is its double SHA-256 hash. Then, the next item in the input is the index of an output in the previous

transaction. The first output in a transaction has index 0, the second output has index 1 and so on and so forth, just like in the example that we saw.

These first two items, they collectively identify the particular output in a particular transaction which will be spent as part of this regular transaction. And the third input in the regular transaction is a response script which satisfies the conditions required to spend the bitcoins in the output. So, this response script includes the digital signature corresponding to the challenge script in the output which is spent as part of this input. Each input unlocks all the bitcoins associated with the output of a previous transaction, whether it is coinbase or regular. So, we mentioned this earlier.

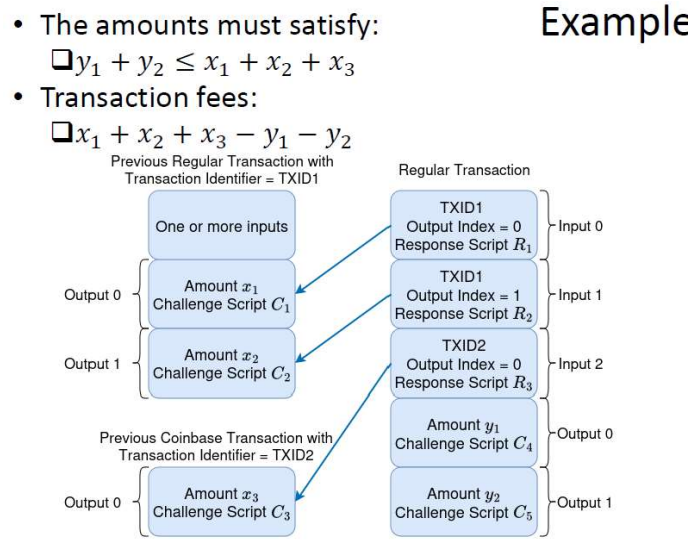
The outputs of a regular transaction have the same format as the outputs of a coinbase transaction. So, we have discussed the format earlier. Each output has an amount and a challenge script. So, each output specifies the amount of bitcoins being associated with that output and a challenge script. The amounts in the regular transaction outputs can take any value such that the sum of the amounts does not exceed the total amount of bitcoins unlocked by the inputs of the transaction.

- Suppose a transaction has N inputs and M outputs
  - let i'th input unlock x_i bitcoins from a previous transaction output
  - let j'th output specify an amount of y_j bitcoins
- The transaction is valid if:
  - $\sum_{j=1}^{M} y_j \leq \sum_{i=1}^{N} x_i$
- The difference $\sum_{i=1}^{N} x_i - \sum_{j=1}^{M} y_j$ is:
  - the transaction fees paid to the miner who includes this transaction in a block

As an example, suppose a transaction has N inputs and M outputs. Further, suppose that the i^{th} input unlocks x_i bitcoins from a previous transaction output, and the j^{th} output specifies an amount of y_j bitcoins. In that case, when will the transaction be valid? So, the transaction will be valid if the total bitcoins coming from all the sources, that is all the inputs, is more than or equal to the total outflow, that is the sum of the amounts in all outputs. So, the transaction is valid if $\sum_{j=1}^{M} y_j \leq \sum_{i=1}^{N} x_i$.

So, the total inflow is this much: $\sum_{i=1}^{N} x_i$. This is the total amount that is unlocked from previous transaction outputs. The total amount paid to other nodes as part of the outputs, so that is, the total amount paid to other nodes is this much: $\sum_{j=1}^{M} y_j$. Hence, clearly this must hold. We cannot pay more than the amount that is unlocked by all the inputs.

So, what happens to this difference: $\sum_{i=1}^{N} x_i - \sum_{j=1}^{M} y_j$? This difference is the transaction fee that is paid to the miner who includes this transaction in the block. Out of all the revenue that comes from all the inputs, some of it is paid as transaction fees to the miner who includes this transaction in the block, and the rest is included in different outputs, which are transferred to other nodes subsequently. This shows an example. This is a previous regular transaction with transaction ID TXID1, and this is a previous coinbase transaction with transaction ID TXID2, and this shows a regular transaction.



It has three inputs, input 0, input 1, and input 2, and two outputs, output 0 and output 1. In the first input, that is input 0, it points to this particular output, that is output 0 of the transaction with transaction ID TXID1. Hence, the transaction ID is TXID1, that is, the ID of this transaction and the output index is 0. That corresponds to this output 0 of this transaction with ID TXID1. And the response script is R_1.

Similarly, for input 1, the transaction ID is the same, transaction ID 1. And the output index is 1. So, this one unlocks this particular output. And for this third input, that is input 2, the transaction ID is TXID2, that is it corresponds to this coinbase transaction, which has a single output. And the output index is 0.

So, it points to this particular output, the only output of this coinbase transaction and the response script is R_3. So, these are the three inputs of this particular regular transaction and there are two outputs. One is an output with amount y_1 and challenge script C_4 and one is an output with amount y_2 and challenge script C_5. So, let's consider this regular transaction.

We have seen that the total amount that is inflow to this regular transaction is $x_1 + x_2 + x_3$ and the total outflow is $y_1 + y_2$.

Hence, $x_1 + x_2 + x_3 \geq y_1 + y_2$. Hence, this inequality holds. Typically, $x_1 + x_2 + x_3 - y_1 - y_2$ will be positive, and that will be the transaction fee. So, the transaction fee is this: $x_1 + x_2 + x_3 - y_1 - y_2$. So, out of the inflow, a part is the transaction fees, and the rest is paid to the outputs y_1 and y_2.

So, in summary, we discussed different aspects of the Bitcoin protocol. We discussed the meaning of blockchain forks and we discussed how blockchain forks are resolved. And then we discussed the format of transactions. There are different kinds of transactions. We discussed coinbase transactions and regular transactions.

We'll continue our discussion of the Bitcoin protocol in the next lecture. Thank you.