

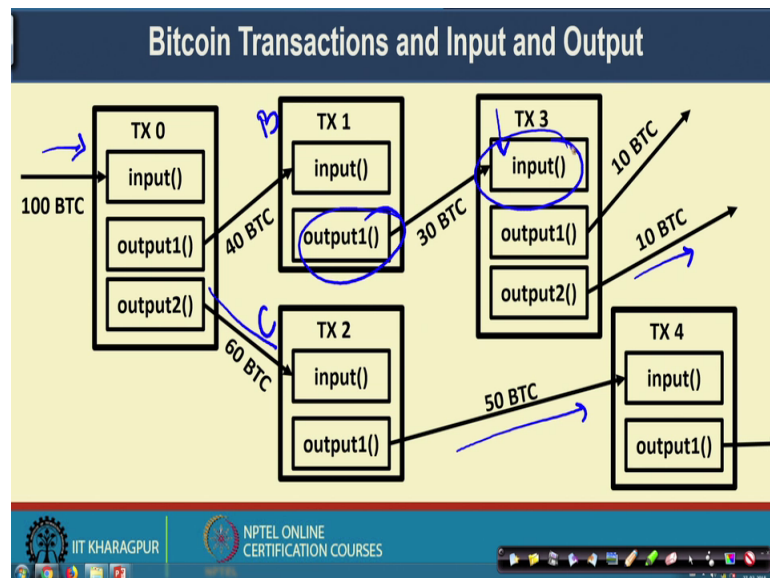
**Blockchains Architecture, Design and Use Cases**  
**Prof. Sandip Chakraborty**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 08**  
**Bitcoin Basics – II**

Welcome back to the course on Blockchain. So, in the last class we have a looked into a little details about the basic bitcoin transactions and how we make a transaction of certain bitcoin from one account that is characterized by a address.

So, from one bitcoin address to another bitcoin address now in this particular lecture we will look into the details of the concept of bitcoin script. So, during the last lecture we have looked into this kind of forth programming language based on which the bitcoin scripts are designed. So, we will look a more details about this bitcoin script.

(Refer Slide Time: 01:00)



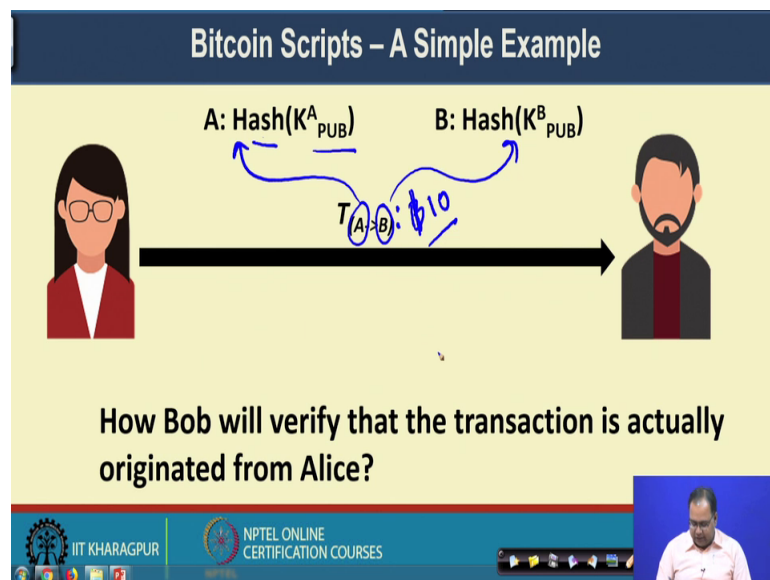
So, as we were mentioning that the transactions in a bitcoin can be characterized by the inputs and the corresponding outputs. So, it is like that the amount of bitcoins, which are being transferred from one user to another. So, you can you can represent the bitcoin transactions in the form of series of inputs and outputs. So, it is like that one user the Alice he has received some 100 bitcoins. So, once alices has received this 100 bitcoins out of this 100 bitcoins Alice sends a say 40 bitcoins to Bob and some 60 bitcoins to Charley. So, in that case this particular transaction has one input which is coming to

Alice from someone else and has two output one output is going to the input of Bob and another output is going as the input of say Charley.

Now, once Bob get this 40 bitcoin Bob by for this 40 bitcoin Bob can spend 30 bitcoin the out of those 40 bitcoin to some say forth person say some delta with 30 bitcoin and the 10 bitcoin are still with Bob. Then this 30 bitcoin they can be spend in one output at 10 bitcoin another output at as another 10 bitcoin and the Charley he had received the some 60 bitcoin out of this 60 bitcoins he can spend some 50 bitcoins in a transaction.

So, that way the bitcoins that are generated in the network the bitcoin moves through multiple parties with this different transactions which are happening. And every transaction that we have mentioned that can be characterized by one output and the corresponding input. Then this output of a transaction is like someone is sending that particular bitcoin and the input is that someone is receiving that bitcoin. And with the help of the bitcoin script we want to actually ensure that this particular input corresponds to this particular output.

(Refer Slide Time: 03:16)

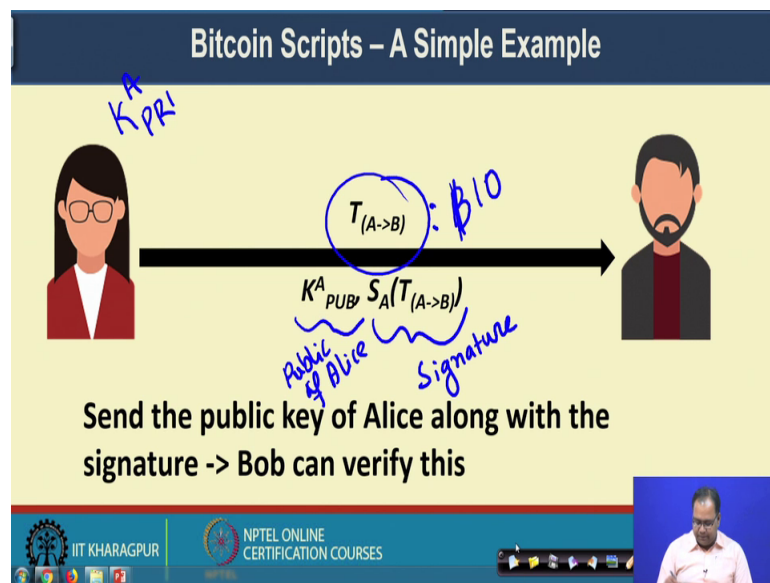


So, let us see that how bitcoin script works. So, explaining the bitcoin script would be easier if we go with an example. So, let us go with an example that Alice wants to make a certain transaction to Bob. So, Alice initiates a transaction that transaction contains the address of Alice and Bob with say certain amount. So, the amount I have not written here say the amount can be something like something likes say 10 bitcoins. So, Alice wants to

make a transaction of 10 bitcoins to Bob and this A and B corresponds to the addresses and as we know these addresses comes from the hash value of the corresponding public addresses.

Now, the question is that how Bob will verify that the transaction actually originated from Alice. So, if the transaction actually originated from Alice and this is not a force transaction then Bob can spend this 10 bitcoin into another transaction.

(Refer Slide Time: 04:09)



So, what we generally do that we have mentioned several times earlier like with the transaction we include two other parameters one is the public key of Alice. So, this is the public key of Alice and we also include the signature from the signature from Alice. So, this two things are included in the transaction; now as we know based under digital signature mechanism that only Alice can have her private key.

So, Alice has her private key and with these private keys she has generated this particular signature and if the public keys of Alice is available using this public key, you can verify the signature. So, Bob can verify this signature and Bob can be sure that this transaction has actually originated from Alice and that is the valid transaction.

So, in that case the bobs wallet can consider that whatever amount. So, earlier it was some 10 bitcoin the 10 bitcoin amount that has been transferred. So, Bob can Bob can certainly take this 10 bitcoins and use that 10 bitcoins for another transaction.

(Refer Slide Time: 05:21)

**Bitcoin Scripts – A Simple Example**

$T_{(A \rightarrow B)}$

$K^A_{PUB}, S_A(T_{(A \rightarrow B)})$

**Bitcoin indeed transfers scripts instead of the signature and the public key**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now what bitcoin actually does that instead of sending this individual public key and individual public key and corresponding signature bitcoin actually transfer some scripts. So, what are those scripts?

(Refer Slide Time: 05:40)

**Bitcoin Scripts – A Simple Example**

$T_{(A \rightarrow B)}$

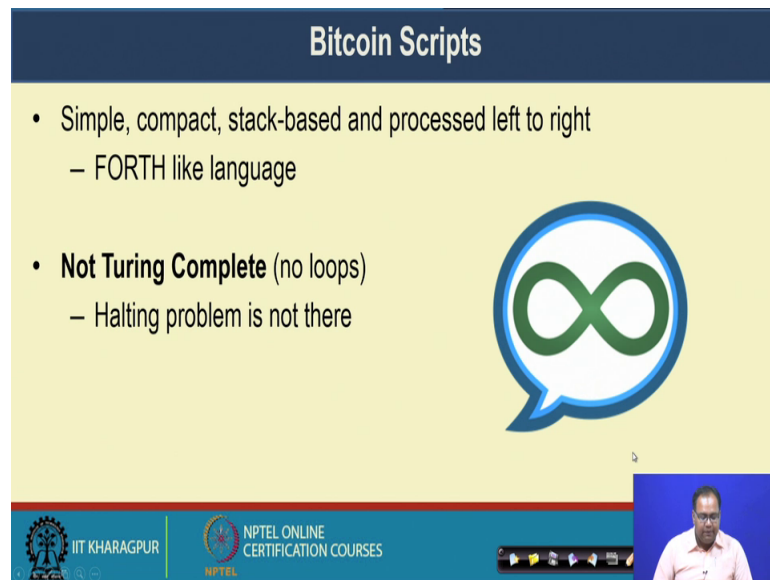
*scriptSig, scriptPubKey*

**Bitcoin indeed transfers scripts instead of the signature and the public key**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, traditionally bitcoin uses two scripts one is corresponds to the signature which we call as the scriptSig and another script corresponds to the public key that is the script public key. So, this script public key basically tells you that how you can use the pub key to validate the corresponding signature.

(Refer Slide Time: 06:00)



The slide is titled "Bitcoin Scripts" in a dark blue header. The main content area is light yellow and contains two bullet points: "Simple, compact, stack-based and processed left to right" with a sub-bullet "– FORTH like language", and "Not Turing Complete (no loops)" with a sub-bullet "– Halting problem is not there". To the right of the text is a large green infinity symbol inside a blue speech bubble. The footer is dark blue and contains logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a speaker.

So, this bitcoin script in general it is a simple programming language which is compact a stack based language which is processed from left to right. So, from left operator or operand you can start processing it and then you can proceed further in the right and it is a kind of forth like language. So, it was inspired from forth and based on the forth language; the basic bitcoin script has designed. But the interesting fact of bitcoin script is that it is not turing complete; it means like it does not support all the operations like it does not support a loop kind of operation and it was intentionally made simple ah

So, that people can the write down some simple scripts and the scripts can be validated and it also makes difficult to right down some kind of attack scripts. And because this particular scripting language is not turing complete by turing completeness we mean that it supports any of the language that can be accepted by a turing machine; because we do not have this concept of loops and few other complicated things we this particular scripting language was not traditionally turing complete. But recently the bitcoin developers they are including new features in the script and they are planning to make it turing complete language, but keeping that side apart intentionally the language is very simple and as it is not a turing complete language you do not have the kind of halting problem.

So, it does not have the loop; so, you do not have a problem of infinite loop inside your language.

(Refer Slide Time: 07:46)

**Bitcoin Scripts**

- With every transaction Alice must provide
  - A public key that, when hashed, yields the address of Alice embedded in the script
  - A signature to provide ownership of the private key corresponding to the public key of Alice

Handwritten diagram: A circle labeled 'A' has an arrow pointing to 'B'. Below 'A' is 'K A' and below 'B' is 'PUB'. A curved arrow labeled 'Hash' points from 'A' to 'B'. To the right, the word 'Signa' is written.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, so, in case of bitcoin script. So, with every transaction Alice must provide a public key. So, this public key whenever you are making hashing over that public key it yields the address of Alice which is embedded inside the key inside the script; that way you can validate that the transaction was actually originated from Alice. And then a signature that signature verifies that the public key which has been transferred that public key actually corresponds to Alice.

So, the idea is very interesting the idea is that you are making a transactions from say A to B along with this transaction. So, Alice to Bob along with that an transactions you are putting a public key say the public key of say the public the public key of Alice and you are putting the signature of Alice which was generated using the private key of Alice.

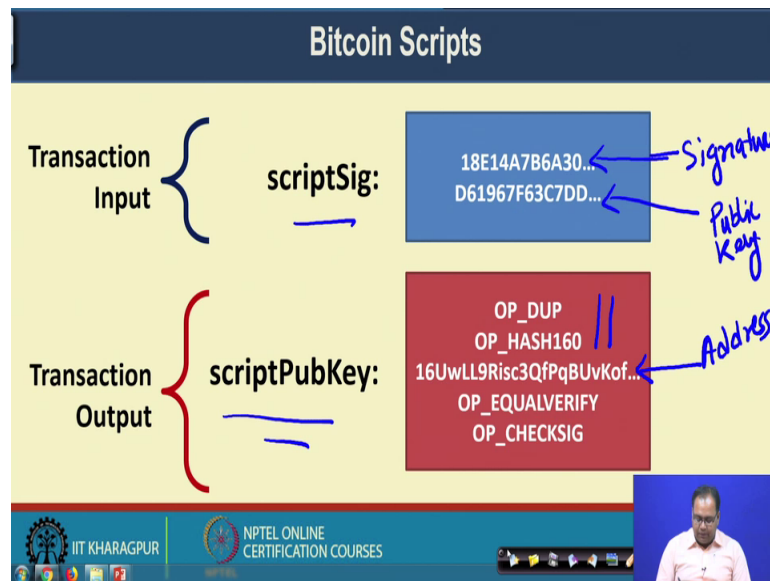
Now, what is the guarantee that the public key that you are getting that public key actually belongs to Alice? So, what you can do first of all you can using this public key you can check the signature if the signature matches; that means, you actually know that this private key what was what has been used for generating this particular transaction; it actually belongs to belongs. So, this public key belongs corresponds to the private key which was used to generate this signature.

Now, as you know this particular address was actually generated from this public key by applying some hash function. So, you can apply the same hash function over this public key and you can see whether you are getting back this address or not. If you are getting

back this address; that means, that this was the original sender who had actually sent the transaction to you and by doing this validation you are making sure that the transaction is actually authenticated and no one else in the future will be able to claim the transaction. And at the same time later on Alice will not be able to deny about this particular transaction.

Now, these things are actually embedded inside the script. So, let us look into this example script.

(Refer Slide Time: 10:13)



So, as you have mentioned that we have the transaction input and the transaction output. So, at the transaction input; we have this scriptSig which contains the signature. So, this signature corresponds to; so, so we have two component here one the first component is signature, it is the signature of Alice and the second component is the public key.

So, we have this components; so, we have the signature and the public in this scriptSig and in script public key you have certain operation and then you have the address of Alice. So, that hash function where that the hash through which you are actually getting out the address. So, this address is embedded inside the script; now let us look into that how this script are actually executed and a using this script sig and script pub key how can you validate the authenticity of the transaction.

So, this is a sample script; so, in the sample script you have two entries you have the script pub key and a script sig.

(Refer Slide Time: 11:30)

**Bitcoin Scripts**

```
scriptPubKey: OP_DUP OP_HASH160  
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

```
scriptSig: <sig> <pubKey>
```

- The stack is initially empty. Both the scripts are combined – input followed by output

```
<sig> <pubKey> OP_DUP OP_HASH160  
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

. So, in the scriptSig you have the signature and the public key and in script public key you have certain operations. So, while explaining the operation I will actually explain you what is the meaning of all this different operations. So, as you have mentioned earlier that bitcoin script is a forth like a programming language. So, it is processed from left to right and it is processed with the help of a stack. So, will take a help of a stack and try to process this particular script.

So, let us look into that how this script is being processed. So, initially the first task is that we need to combine these two scripts together. So, by combining this two scripts together we get a combined script like this. So, first we put the input and then we put the output. So, the input script is your scriptSig. So, first you put the scriptSig that mean the sig and the public key and then you put your script pubkey that is the output script. So, the output script is written here; so, initially the stack is empty and you have generated this combined script by combining the input script and the output script.



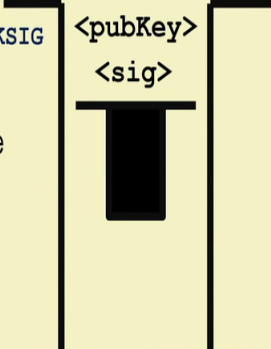
(Refer Slide Time: 12:42)

### Bitcoin Scripts

```
<sig> <pubKey> OP_DUP OP_HASH160  
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

- The stack is initially empty. Both the scripts are combined

```
OP_DUP OP_HASH160 <pubKeyHash>  
OP_EQUALVERIFY OP_CHECKSIG
```



The diagram shows a vertical stack with two items: <pubKey> at the top and <sig> below it. A horizontal line is drawn above the <sig> item, and a vertical line is drawn to the right of the stack, indicating the stack's boundary.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, once you have done it then let us process it using the similar concept of this reverse polish notation that we had used to process forth language. So, how will you process that? We have this initially sig and pub key which are the operand. So, if the as these are the operand we push that operand in the stack.

So, after pushing the operand in the stack initially the stack was empty then we have pushed these two operand in the stack. Now that is our resultant script after pushing this to operand in the stack.

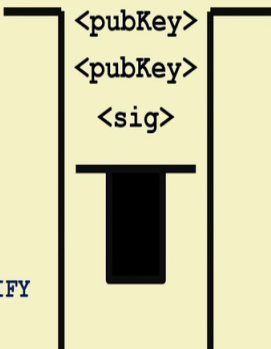
(Refer Slide Time: 13:17)

### Bitcoin Scripts

```
OP_DUP OP_HASH160 <pubKeyHash>  
OP_EQUALVERIFY OP_CHECKSIG
```

- Top stack item is duplicated

```
OP_HASH160 <pubKeyHash> OP_EQUALVERIFY  
OP_CHECKSIG
```



The diagram shows a vertical stack with three items: <pubKey> at the top, another <pubKey> below it, and <sig> at the bottom. A horizontal line is drawn above the bottom <sig> item, and a vertical line is drawn to the right of the stack, indicating the stack's boundary.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, then we have this operator OP dup. So, what OP dup does OP dup the dup operator it actually makes a duplicate of the stack top value. So, earlier the stack dup stack value was this pub key; so, on top of that pub key we have duplicated that things. So, another entry of that pub key is pushed into the stack by this OP dup operator then we have this.

(Refer Slide Time: 13:41)

The slide, titled "Bitcoin Scripts", illustrates the execution of a script. The script consists of three operations: `OP_HASH160`, `OP_EQUALVERIFY`, and `OP_CHECKSIG`. The stack initially contains `<pubKeyHash>`, `<pubKey>`, and `<sig>`. The `OP_HASH160` operation hashes the top stack item (`<pubKey>`) using RIPEMD-160, resulting in a new `<pubHash>` being pushed onto the stack. The stack now contains `<pubHash>`, `<pubKey>`, and `<sig>`. The `OP_EQUALVERIFY` operation compares the `<pubHash>` with the `<pubKey>`. The `OP_CHECKSIG` operation verifies the signature `<sig>` against the `<pubKey>`. The final stack state is `<pubKeyHash>`, `OP_EQUALVERIFY`, and `OP_CHECKSIG`. A blue circle highlights the `<pubHash>` entry in the stack. A blue line underlines the text "Top stack item is hashed (RIPEMD-160 hashing)".

So, what this hash 160 operator does? It applies this RIPEMD 160 hashing mechanism on top of the. So, it applies this RIPEMD 160 hashing mechanism on top of this pub key entry that was there at the top of the stack. So, by applying this hash you are getting a operand like pub hash.

So, after applying this operator you got the pub hash and that is pushed inside the stack; then at the next step ok.

(Refer Slide Time: 14:21)

**Bitcoin Scripts**

`<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG`

- The constant is pushed in the stack

`OP_EQUALVERIFY OP_CHECKSIG`

Stack:  
- `<pubKeyHash>`  
- `<pubHash>`  
- `<pubKey>`  
- `<sig>`

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

At the next step operator of hash 160; you have the pub key hash. So, you this is the operand; so, you push the pub key hash in the stack top then you have the operator equal verify.

(Refer Slide Time: 14:33)

**Bitcoin Scripts**

`OP_EQUALVERIFY OP_CHECKSIG`

- Equality is checked between the top two items in the stack

`OP_CHECKSIG`

Stack:  
- `<pubKeyHash>`  
- `<pubHash>`  
- `<pubKey>`  
- `<sig>`

Handwritten diagram: A circle containing 'A' with an arrow pointing to 'B'.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this equal verify checks whether the two hash values which are there whether they are equal or not. If they are equal then you returned true you just simply popped it out and if they are not equal then you return a false or an error. So, this equality is checked

between the top two items in the stack. So, if these two hashes are equal then ah; that means, the ID that has been transferred in the transaction.

So, whenever you have made a transaction from A to B you are actually verifying that the address corresponds to the address of Alice. So, once that thing is done then the next operator is OP\_CHECKSIG.

(Refer Slide Time: 15:23)

The slide is titled "Bitcoin Scripts" and illustrates the OP\_CHECKSIG operation. On the left, the text "OP\_CHECKSIG" is shown in red. Below it, a bullet point states: "Signature is checked based on the top two stack item". At the bottom left, the word "TRUE" is displayed. On the right, a stack diagram shows two items: "<pubKey>" on top and "<sig>" below it. A black rectangular block is positioned between the two items, representing the OP\_CHECKSIG operator. The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a small video inset of a presenter.

In the OP\_CHECKSIG operator, it again takes the two operands from the stack top and after taking these two operands from the stack top; it verifies that whether the signature corresponds to this public key whether that is a valid signature or not. So, you are doing a signature validation here; so, after doing a signature validation here if you have validated that the signature is correct; that means, you have validated that the signature is actually coming from Alice.

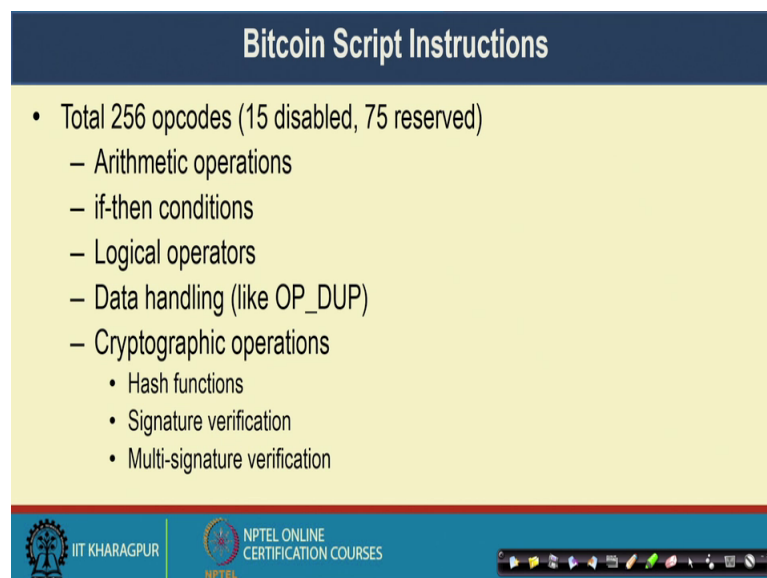
So, in total what you have basically done by executing the input script and the output script by looking into the output script; you have found out that well Alice actually made that particular transaction. So, the address which was there inside the transaction that address corresponds to the address of Alice.

And then by looking into the output script where they are you have the signature and the public key; you have actually validated that that the signature corresponds to that particular public key; so the signature is a valid signature of Alice. So, the combining this

input script and the output script together you have actually ensured that the transaction has originated from Alice itself and the transaction is a valid transaction.

So, now once you have validated the transaction that the transaction is a valid transaction then you can include that transaction accept that transaction as a correct transaction and whatever bitcoin that has been transferred to Bob; Bob can start utilizing that amount of bitcoin for other transactions. So, this bitcoin script it is an very intelligent mechanism to make such kind of checking that whether a particular transaction output sorry a transaction input; it corresponds to a transaction output or not.

(Refer Slide Time: 17:23)



The slide is titled "Bitcoin Script Instructions" and lists the following categories of opcodes:

- Total 256 opcodes (15 disabled, 75 reserved)
  - Arithmetic operations
  - if-then conditions
  - Logical operators
  - Data handling (like OP\_DUP)
  - Cryptographic operations
    - Hash functions
    - Signature verification
    - Multi-signature verification

The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a navigation bar with various icons.

Now, in case of bitcoin instruction that was the earlier version of bitcoin script the initial version of bitcoin script it has the total of 256 different opcodes out of them 15 are currently disable and 75 per reserved. So, you have different arithmetic operators, you have different if then conditions then the logical operators, the data handling operators like OP dup which duplicate the entries of a stack top.

Then you have different kind of cryptographic operators like the hash functions, the signature verification operations, the concept of multi signature verification operator operations where the signature from multiple users are combined together and that multi signature is used to verify the signature coming from a set of users. So, you have all those different kind of OP codes as a part of your bitcoin script.

(Refer Slide Time: 18:17)

**Interesting Bitcoin Scripts**

- **Provably un-spendable or prunable outputs**  
`scriptPubKey: OP_RETURN {zero or more ops}`  
*false*
- **Anyone-can-spend outputs**  
`scriptPubKey: {empty}`  
`scriptSig: OP_TRUE`

Source: <https://en.bitcoin.it/wiki/Script>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, here are some interesting bitcoin script; so, the first bitcoin script it talk about that you are making some kind of transactions for probably un spendable or prunable output. So, in the script pub key mistakenly you are writing as OP return. So, whenever you are immediately writing as OP return; so it will immediately return from the stack top with the false operand.

So, whenever it will return from the stack top with a false operand then you are sure that this particular transaction whatever bitcoin that has been transferred as a part of this transactions there are not usable bitcoin. So, you cannot use that bitcoins any further then the second interesting script that anyone can spend output; it is like that whatever transactions you are making in the bitcoin network anyone in the network can use those bitcoin.

So, in that case in the pub key we have make it empty. So, you do not have any pub key information in the scriptSig it is always returning as true. So, who whoever is going to validate that the script they will immediately get a true and they will be able to use that bitcoin so, without making any further validation.

(Refer Slide Time: 19:42)

**Interesting Bitcoin Scripts**

- Freezing funds until a time in the future

```
scriptPubKey: <expiry_time>
OP_CHECKLOCKTIMEVERIFY OP_DROP OP_DUP
OP_HASH160 <pubKeyHash> OP_EQUALVERIFY
OP_CHECKSIG
scriptSig: <sig> <pubKey>
```

Source: <https://en.bitcoin.it/wiki/Script>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Then the third script it is like we are trying to freeze the funds until a time in the future. So, in the script pub key you have an you have an expiry time; so, this expiry time specifies that till what time you will be able to you will not be able to use that particular bitcoin. So, this OP check time check lock time lock time verify this particular operator checks that whether the current time is more than the expiry time or not if the current time is less than the expiry time then immediately it will return a return this OP drop; that means, it will return a false and that particular bitcoin will not become usable.

So, if it if this operator check lock time verify if it finds out that well the current time is more than the expiry time; that means, the time has been expired; then it will go to this next part of the operator, it will skip this OP drop it will come to this normal validation script which was OP dup followed by OP hash 60 the pub key hash equalverify and check sig the normal script it will execute. And the intended participant will be will be able to use that fund in future.

So, this particular script basically ensuring that the bitcoin that has been transferred that bitcoin will remain unusable for certain amount of duration; it is like that you have transferred certain bitcoin, but you do not want the user to be able to use the bitcoin say for three days that you can embed inside this kind of scripts.

(Refer Slide Time: 21:30)

**Bitcoin P2P Network**

- An ad-hoc network with random topology, Bitcoin protocol runs on TCP port 8333
- All nodes (users) in the bitcoin network are treated equally
- New nodes can join any time, non-responding nodes are removed after 3 hours

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

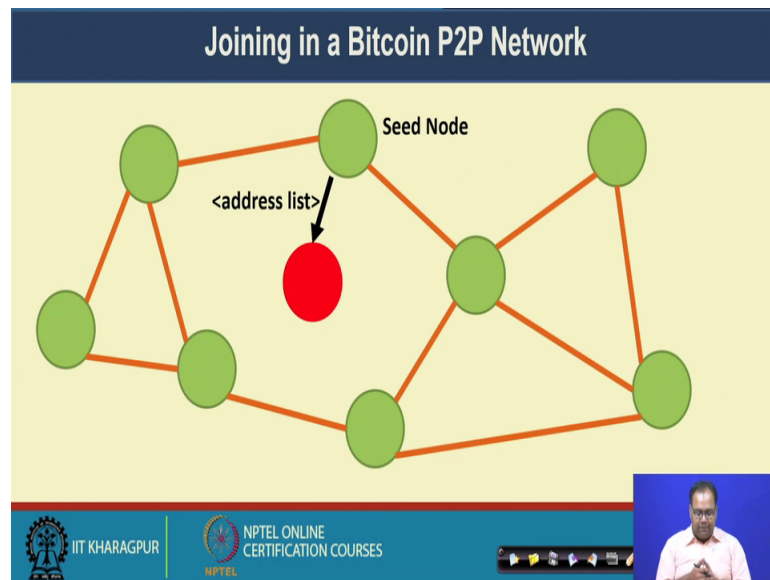
Well next we come to the concept of bitcoin peer to peer networks. So, as you have discussed earlier that bitcoin works on top of a peer to peer network architecture in a complete decentralized manner. So, this bitcoin peer to peer network it is a adhoc network with certain kind of random topology and on top of that it runs the bitcoin protocol using the well known TCP port 8 triple 3. And all nodes who are the users in that bitcoin network they are treated equally and new nodes can join in the bitcoin network any time and one interesting property of bitcoin is that it is permission less.

So, to join in the bitcoin network you do not need to verify yourself to some authentication server that is not required. So, whenever you want to join in the bitcoin network; you can simply open up you are wallet and after opening up your wallet you can immediately start send a join message to join in the network can start doing your own transactions.

Now, for the old nodes if the old nodes are not responding the old peers; if they are not responding for certain duration in generally it is for three hours then they are automatically removed from the bitcoin network. And those users whenever they will open their wallet again and start participating in some transaction; they can join in that specific network.



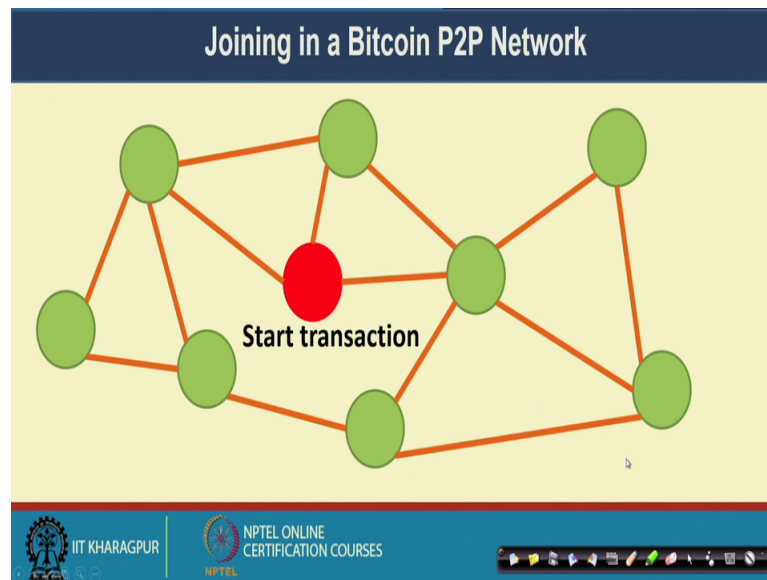
(Refer Slide Time: 22:55)



Now, let us look into that how you can implement joining in a bitcoin peer to peer network. As we have said that in a bitcoin network it is a kind of a random topology of nodes and the leaves that are there in between two nodes those are kind of virtual link or those are kind of overly link. So, do not expect that there are kind of physical links between the two users; those are kind of virtual links or a kind of peer to peer link; that means, we are considering if the if I have a link with say another user Alice so; that means, from myself to Alice we are virtually considering each other as a neighbor.

So in such a kind of overly network whenever a new node comes and once to join in the network; so, there are certain nodes in the network which work as the seed node. So, the task of the seed node is to provide the initial information to the new nodes who are going to join in the network. So, whenever you are installing your bitcoin applet inside your bitcoin applet the address of those seed nodes are already there. So, whenever you are trying to join in the network you send a message to one of the seed node that give me the peer addresses. So, in response this seed node will send you a set of addresses whom you can consider as the peer.

(Refer Slide Time: 24:19)



Now, among that set of addresses which has been returned by the seed node; you can select certain random number of nodes and you can create a kind of peering relationship with those particular nodes and you can join in this overly network.

So, this means that whenever you have joined this overly network whatever information you are going to send in the network. So, as you already know that whenever you are making a new transaction; the new transactions are broad casted in the network or for the mining node, whenever the mining node has found out some new blocks the new blocks are again broad casted in the networks and whenever you have done. So, then you can you can broad cast those transactions or broad cast the new block information to your peer nodes

So, once you have joint in the network the first task of a node is to get the most recent blockchain from your peer nodes and update your local copy of blockchain. So, that is again a interesting principal like you may be offline for certain duration and whenever you are offline; during that time maybe the blockchain has increased with the new transactions and it will certainly happen whenever that has happened then you are you are actually after doing this joining procedure, you are asking your peers to give them the most recent or most updated blockchain information that they have. So, once they are transferring the most recent blockchain information to you compare those blockchain.

So, here we apply some kind of 50 percent rule because of this principle of decentralization it may sometimes happen. So, the details will be discussed later on it may sometimes happen that you are getting two different copies of blockchain from your neighbours. If you are getting two different copies of blockchain from your neighbours you accept the copy of the blockchain which has been transferred to you by most number of peers; that means, if a certain copy of the blockchain has been transferred to you by more than 50 percent of the nodes in your peer list; then you accept that particular copy of blockchain.

And while accepting the blockchain copy; you also need to need to need to ensure that you are also considering the longest chain in the current blockchain. So, once you have got the most recent copy of the blockchain then you can start the transaction procedure. So, that is all about today's lecture. So, in the next lecture we will look into the details about after doing this joining procedure; how can you initiate a transaction and how that transaction gets propagated in the bitcoin network. So, see you again.

Thank you.