

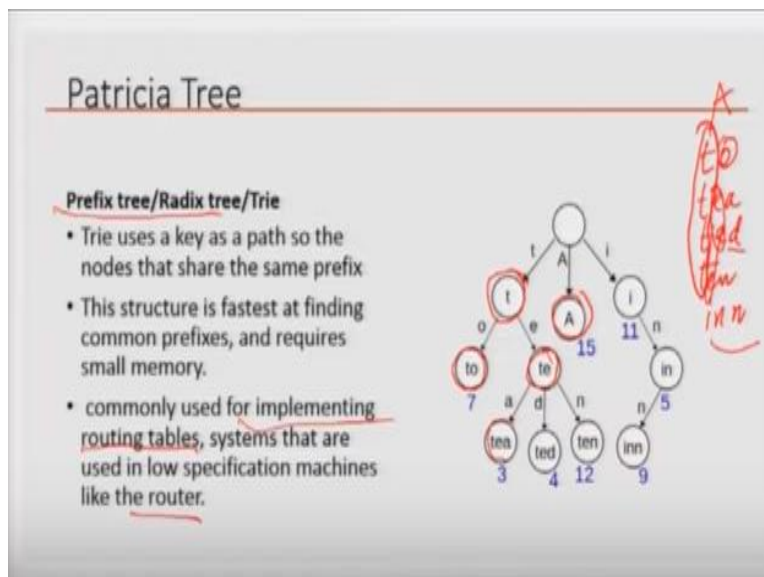
**Introduction to Blockchain Technology & Applications**  
**Prof. Sandeep Shukla**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology-Kanpur**

**Lecture - 16**

Welcome back to Blockchain Technology and Applications, another lecture. So last time we were talking about Ethereum blockchain and what are some of the differences between Ethereum and bitcoin and one of the difference we saw is the difference between how the state is maintained in the blockchains in case of Ethereum.

It is based on accounts and in case of blockchain it is based on UTXOs or you know unspent transactions. Now the question, remember that in case of the block, the how the blocks are kept in the bitcoin we talked about Merkle tree, right.

**(Refer Slide Time: 00:56)**



So in Merkle tree we have a very specific data structure, which basically is a hash tree. Now Merkle trees, quite a bit of work to find a particular transactions for a particular account, for example, in case of bitcoin in case of Ethereum. So they use something called a Patricia Merkle tree. And so Patricia tree is a data structure, which is actually based on the idea of a prefix tree or radix tree or trie.

So the idea there is that I want to save space, but I want to record a number of different strings, let us say. Not everything is not necessarily strings. It may be other

objects like account addresses and information like that. So the question is, how do you store them? So in a trie, what happens is that if you have number of different let us say strings.

And you do not want to so if you want to store the strings in a Merkle tree, for example, then if you have, let us say 100 strings, then you will have 100 leaves, each of them will be storing the string, and then two of them will be hashed and then another two will be hashed. So you will have a height of log of 100, which would be about height of let us say, log of 100 will be about close to maybe log to the base 2 is 7.

So you might have a tree of height 7, and then all the different things. Also finding something there unless the leaves are sorted, will be quite a bit of work. You do not know when you start from the root which direction to start. So the idea of trie is that you keep the prefixes of the different, prefixes of the different common prefixes on the path. So for example, suppose I want to store this strings to, tea, ted, ten and inn, right.

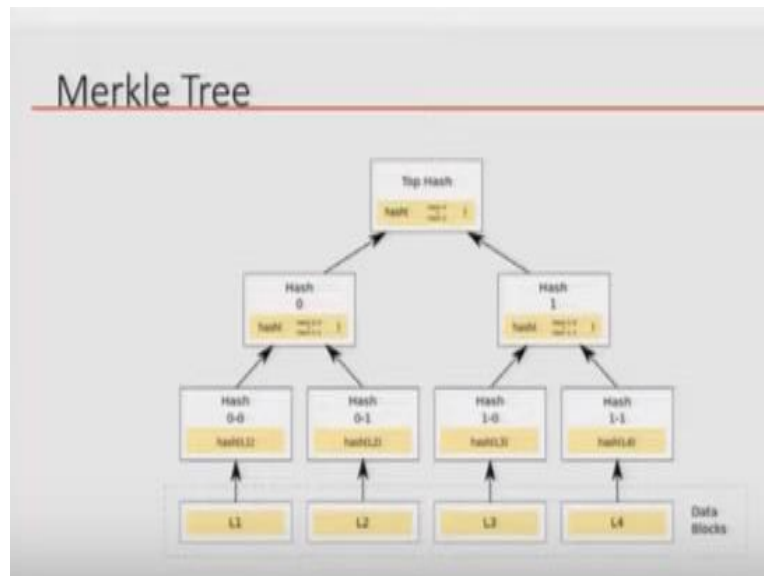
So in this case, what you do is that we see that there are four of the strings starts with the same prefix, so t. So I will create a root and then I will have a child which is marked whose paths is marked with t and then I will have note for t and then I see that after t there are two with e and one with o and actually three with e. So therefore, I will create one branch with e and then it will be te and then I will be easily create this tea, ted and ten.

And then you will have this one which is to that is separate. Similarly, here you will have, let us say there was another word A. So A is alone here and then you will have this one inn was different. So inn then n and in this will stand out separately. So we have basically used the prefix encoding in creating the data structure and therefore, if I want to search ted I know that I have to go in this direction, go in this direction, and then go in this direction.

So the searching will be log n and it will be even sometimes if you have much more commonality in the prefixes it will be even easier. So this data structure trie is used

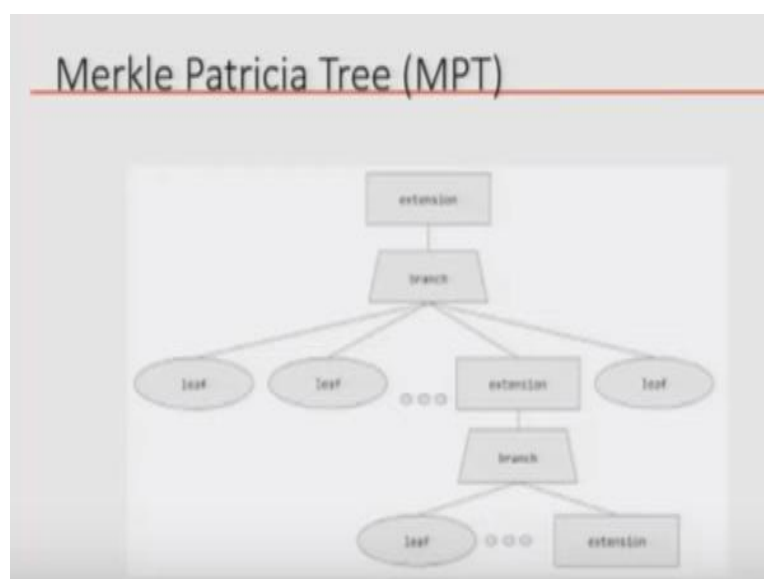
for implementing for example, routing table and routers and so on. So this is not something new that is being invented here. But they decided that this is how they are going to store their account informations.

**(Refer Slide Time: 04:53)**



And then Merkle tree you already know. So there is nothing new to discuss in Merkle tree. So if you want to keep different data, then make them the leaves and then you start making hashes and get the hash. The root hash will give you integrity of the entire thing.

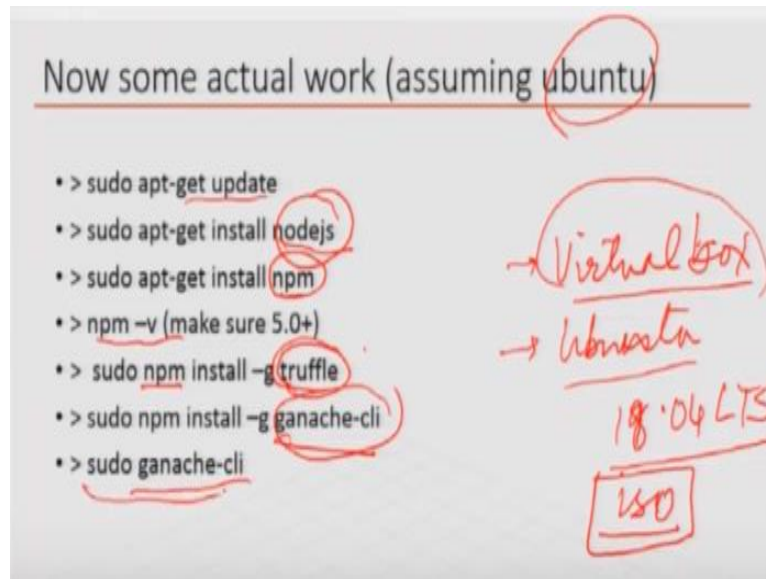
**(Refer Slide Time: 05:11)**



So they combined the idea of Merkle Patricia tree as the data structure through which the blocks are stored. So since we are not going to really work on the blocks, per se, I

just want to give you an idea that this is another different optimizational aspect of Ethereum compared to the bitcoin blockchain.

**(Refer Slide Time: 05:36)**



So now what I want to do is I want to tell you how to go about playing with Ethereum blockchain and creating your own network so that you are not initially having to work with Ropsten or the or certainly not on the Mainnet. You create a simulated Ethereum blockchain on your machine using something called ganache, and then you use truffle, which will be used for creating smart contracts and applications and the ability to deploy contracts.

So what these things will give you is the ability to very quickly get started with creating smart contracts. For example, you want to create some smart contract like already skeleton smart contract available, let us say and then you want to fill that in a little bit to create some functions in the smart contracts and play with it. So what I would do is that I will assume you have an Ubuntu.

Otherwise, what you can do is that you can download a VirtualBox from the from Oracle. It is a free software, VirtualBox. You can Google it, and then you can download it. It is basically a virtual machine, ability to host a virtual machine through VirtualBox. Then Ubuntu distribution sites. And you can download Ubuntu let us say 19 or 18.04 LTS whatever is the version you want to get. And you get a ISO image.

It is going to be pretty heavy, couple of gigabytes. But nowadays that is not a big deal. So you download that and then you create a guest machine in the VirtualBox with Ubuntu. And then you can use that as your experimentation machine within your let us say Windows machine. Or what you can do is that you can find the instructions for creating nodejs and installing nodejs and truffle and ganache-cli on your Windows machine.

I have not done that. But I am pretty sure you can find how to do that on the internet. So but I am going to assume that you have Ubuntu. So you have to update your Ubuntu first and then install nodejs. And nodejs is actually a development environment for JavaScript applications. And then you do a install the node package manager, which will allow you to do various things like install truffle, install ganache-cli and so on.

You want to check the version of npm. It is 5.0 plus is good. Then you use the package manager npm, to install three things, truffle, two things truffle and ganache-cli. And then you ganache-cli will also create an account ganache-cli. So you actually do this.

**(Refer Slide Time: 08:47)**



Now when you start ganache-cli, right so you will see that it basically is starting to emulate Ethereum blockchain, which has already reconfigured 10 accounts 0 to nine and their addresses are given here. You may actually record this copy this addresses

to a file and the corresponding private keys. So these are the private keys. And then you have the gas price here. This is in terms of Wei and then there is a gas limit.

But this gas and gas price and all that stuff is going to be fake because this is not against the Mainnet so you do not really have to pay any money and then it will start listening for nodes to do things at this port 8545. So this port so when you make a when you deploy a transaction, it has to refer to this host and this port, so that you can you know this it goes to this Ethereum emulation, and then you will see the effects of deploying a contract or testing a contract and so on.

**(Refer Slide Time: 10:08)**



```
HelloWorld smart contract
• > mkdir SmartContractDemo
• > cd SmartContractDemo
• > mkdir HelloWorld
• > cd HelloWorld
• > truffle init
```

So then in order to, then you open a different window, because that window will now be busy listening to for, you know nodes to deploy contract or do transactions there. So you make another window and then you create a Demo directory. You make a directory name and go to that directory and then say, truffle init, right. So what truffle init will do is that it will create multiple different directories.

And these directories are called contract migrations test and so on. And these directories will have some skeleton code for the contract called HelloWorld. It does not know what you want to do with HelloWorld. So that is what you have to fill in. But at least you will it will put out the skeleton.

**(Refer Slide Time: 10:58)**

### Truffle init

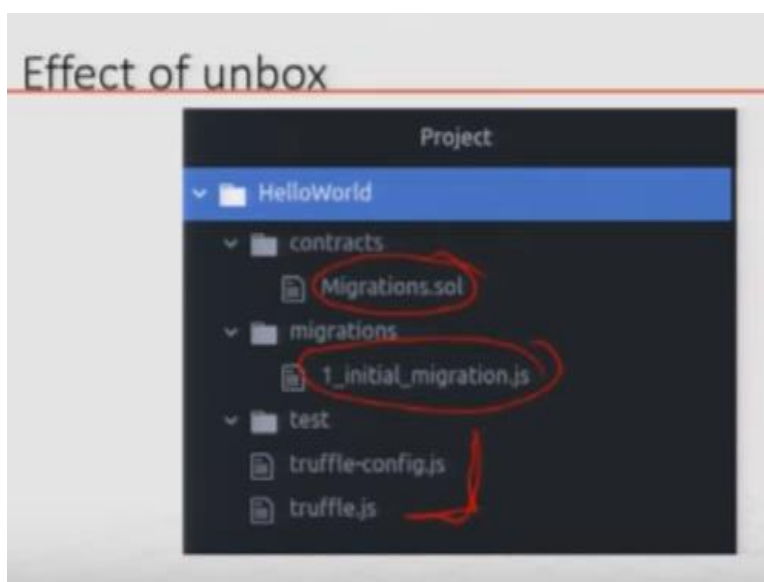
```
william@william-VirtualBox:~/ethereum$ mkdir SmartContractDemo
william@william-VirtualBox:~/ethereum$ cd SmartContractDemo/
william@william-VirtualBox:~/ethereum/SmartContractDemo$ mkdir HelloWorld
william@william-VirtualBox:~/ethereum/SmartContractDemo$ cd HelloWorld/
william@william-VirtualBox:~/ethereum/SmartContractDemo/HelloWorld$ truffle init
Downloading...
Unpacking...
Setting up...
Unbox successful. Sweet!

Commands:
  Compile: truffle compile
  Migrate: truffle migrate
  Test contracts: truffle test
william@william-VirtualBox:~/ethereum/SmartContractDemo/HelloWorld$ ls
contracts  migrations  test  truffle-config.js  truffle.js
william@william-VirtualBox:~/ethereum/SmartContractDemo/HelloWorld$
```

So as you can see here, this is what we are doing. And then truffle can also be used to compile contracts, it can also use to migrate contracts which basically leads to deployment. And then you can do the testing of the contract. And once you have run the truffle in it, you will get this directories under HelloWorld contracts where the smart contract code will be there. Migration is the migration code for deployment.

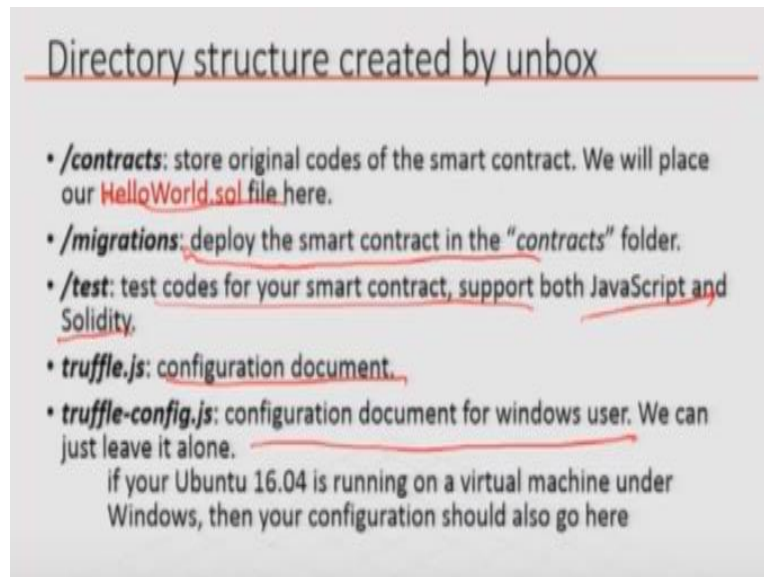
Because when you create a smart contract, you also have to create JavaScript or whatever to actually deploy the contract or call functions on the contract and so on. There is a test directory and there will be configuration JavaScript files. And now you have to start working with this.

**(Refer Slide Time: 11:50)**



So if you look at this thing, then you will see that you have HelloWorld and the HelloWorld is has contracts in which it will have one smart contract for migration called Migrations.sol. And then you have a JavaScript in the migrations directory, which is for the initial migration. And then you have this configuration, other stuff.

**(Refer Slide Time: 12:16)**



So contracts is where you store the Solidity code of the smart contracts. Now we will create a HelloWorld.sol file, and put in the contracts and fill it up. Migrations is also another smart contract that is going to be used to deploy this contract. And test is a test code and it can support both JavaScript and Solidity. And truffle.js is a configuration document. And truffle.js actually is already filled with everything commented.

So you have to uncomment the configuration of the network. And in your case, the network is listening on local host. That is 127.0.0.1 and then it is actually listening to port 8545. So you have to make sure that you uncomment that network configuration in which the port is 8545 in this file, this file you can is only for Windows users or old Ubuntu users. So you can ignore it.

**(Refer Slide Time: 13:22)**



## Creating contract

• > truffle create contract HelloWorld

```
William@William-VirtualBox:~/ethereum/SmartContractDemo/HelloWorld$ truffle create contract HelloWorld
William@William-VirtualBox:~/ethereum/SmartContractDemo/HelloWorld$ cd contracts/
William@William-VirtualBox:~/ethereum/SmartContractDemo/HelloWorld/contracts$ ls
HelloWorld.sol  migrations.sol
William@William-VirtualBox:~/ethereum/SmartContractDemo/HelloWorld/contracts$
```

So now you do this create contract HelloWorld. And then it will create a skeleton for the HelloWorld.sol, which will have the skeleton code where you have to fill things in.

**(Refer Slide Time: 13:34)**

## HelloWorld.sol

```
pragma solidity ^0.5.0;
Contract HelloWorld {
    function hi() public pure returns (string memory) {
        return ("Hello World");
    }
}

> truffle compile
```

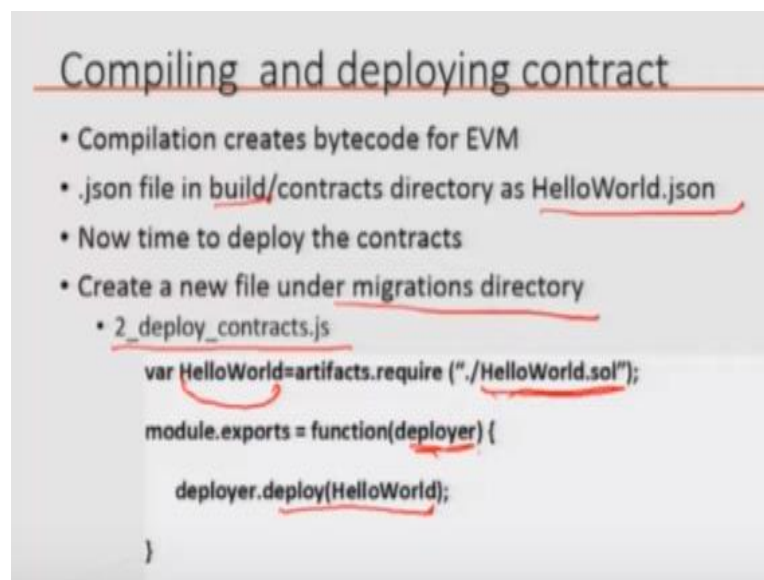
So this will look something like this. So pragma, solidity this says what is the version of Solidity. Solidity is an evolving language in last 4, 5 years. So right now I think it is 0.7. So currently, so this is from last year. So most probably you will see something like 0.7 or something here. And then in the contract HelloWorld you will, there will be a constructor.

Automatically you do not need to do anything in there for this particular example. A constructor is what actually initializes the contract. And it only gets called when you

first time deploy the contract. But if the constructor is skeletal, which basically means that it is not doing anything. But in this case, we want another function, which basically does not take any input, but it returns a string.

And this string is actually is of type memory. So it is not in the storage, it just in the memory, which means that this will not be stored on the blockchain. And then it will return HelloWorld. And then you can call truffle compile, and this will compile the HelloWorld.sol into bytecode.

**(Refer Slide Time: 14:55)**



The slide is titled "Compiling and deploying contract" and contains the following content:

- Compilation creates bytecode for EVM
- .json file in `build/contracts` directory as `HelloWorld.json`
- Now time to deploy the contracts
- Create a new file under `migrations` directory
  - `2_deploy_contracts.js`

```
var HelloWorld=artifacts.require("./HelloWorld.sol");
module.exports = function(deployer) {
  deployer.deploy(HelloWorld);
}
```

Now it also creates a json file, which is in the build. So you will see a build directory that has come up, which was not there when you just created the read the truffle in it. So you will see that json file. In there, there are some of the configuration information. And you do not have to worry about it for this simple contract. And then in the migration directory, you create a another js file, JavaScript file.

Remember there was already one underscore initial migration file. So now you will be creating a 2 underscore initial file. And in that case there you are going to say so it is a JavaScript. So there you are going to say that I have this variable HelloWorld. And the artifacts required for this is the HelloWorld.sol file. And then it will say `module.exports equals function deployer`.

So it requests a deployer and deployer deploys this smart contract. So the name of the smart contract is HelloWorld. So it is going to deploy this. So when you run this

JavaScript, it will deploy based on the configuration information that is already in the truffle.js and so on.

**(Refer Slide Time: 16:17)**



```
15 module.exports = {
16   networks: {
17     development: {
18       host: "localhost",
19       port: 8545
20     }
21   }
22 };
23
```

> truffle migrate

So as I said that in the truffle.js, you have to make this uncomment. So mostly they will come commented like this. So you have to basically uncomment the relevant part, so that you know that your host on which the network host is local host, and then the port at which it is listening is this. And then you call truffle migrate.

And remember, there was a window in which the ganache was running, and it was stuck at the point where you left it. So now as soon as you migrate, you will see that it will tell you what is being deployed and all that stuff. So you will see all these things in the migrate this kind of information the contract address and so on will come in that screen where you have the ganache running.

**(Refer Slide Time: 17:07)**

### Effect of the deployment on the network

```
net_version
eth_accounts
eth_accounts
net_version
net_version
eth_sendTransaction
Transaction: 0x942a4017ab47ebd8ae2d86552656f42775cfb92569d67c2af6b0963d5d4fab
Contract created: 0x84992ef729fbac77a60f5479e48ba70716bb8f6c
Gas usage: 277482
Block Number: 1
Block Time: Fri Jun 22 2018 16:39:21 GMT-0700 (PDT)

eth_sendBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_uninstallFilter
eth_sendTransaction
Transaction: 0x4dc3528ba8ee21ac2e81cac9a2aabe299653fa03000160c80e83e053c4d76c4
Gas usage: 42000
Block Number: 2
Block Time: Fri Jun 22 2018 16:39:22 GMT-0700 (PDT)

eth_getTransactionReceipt
eth_accounts
net_version
net_version
eth_sendTransaction
Transaction: 0x9b1066d86effeba98c11154272d1ae55864fa6d9bc017708ba7859bb43f9e4
Contract created: 0x079d052b39eeFb041d03917b0c0eb71456ba173
Gas usage: 136200
Block Number: 3
Block Time: Fri Jun 22 2018 16:39:23 GMT-0700 (PDT)
```

This is actually the what you will see in the ganache screen where you will see all this different things being invoked. And these are the transactions, block number, and so on and so forth.

**(Refer Slide Time: 17:19)**

- ### Summary
- In this lecture, you learnt the design philosophy of Ethereum Blockchain
  - Some Basic differences between Bitcoin blockchain and Ethereum blockchain
  - The concept of smart contracts
  - The concept of Ethereum network and simulation of such networks for testing
  - Finally, clue about how to simulate a local private network, and try out a simple smart contract on your own

So now you can try this, you can also go to the Ethereum.org website, and you can actually directly play on the, through the web interface. So they have a Hello if you go a little bit down you will see a HelloWorld there will be a coins smart dapp. So various dapps are there. You can also play with that instead of having to do all this yourself. But I think it is more fun to do it yourself than going there.

And because the amount of visibility of what you are doing will be slightly less when you are doing it to the web interface. But you can do that also and you can see number

of different contracts to learn how the contracts look like and how they work and so on. So in this set of lectures, you learnt the design philosophy of Ethereum blockchain, some basic differences between bitcoin blockchain and Ethereum.

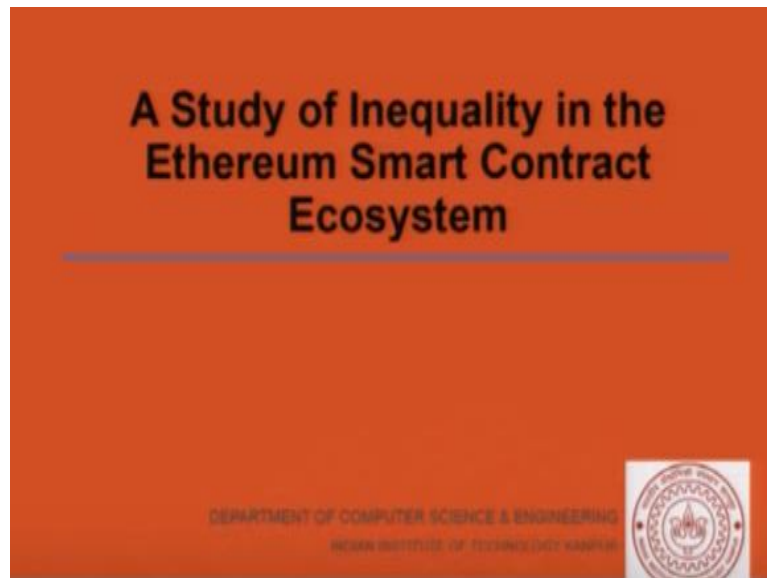
The concept of smart contracts, the concept of Ethereum network and simulation of such network for testing, and finally, a clue about how to simulate a local private network and try out a simple smart contract on your own. So in this course, so we are about four weeks away from the end, and we have to cover a lot of other things.

So next thing that I want to cover is some of the issues that this bitcoin and Ethereum this permissionless and cryptocurrency blockchains have, that we have ourselves at our laboratory have discovered and I want to actually make you aware of those because, as you know probably that RBI had put a restriction on cryptocurrency buying and selling in India.

And then recently in last week, the Supreme Court has asked RBI to enable cryptocurrency exchanges and cryptocurrency selling and buying and so on. And I personally has been very, you know critical of that step because in a country like India where people can be easily fooled and they are not so aware, there are a lot of things about bitcoin and blockchain that people do not know.

And they might actually lose money because it is very speculative. And therefore, I want to give you some little more idea from our own research as to why I think that cryptocurrency is not something that should be bought and sold in India.

**(Refer Slide Time: 20:07)**



So what we are going to do is I want to start talking about this and **and** eventually will talk in details. So first thing that I want to say is that we have studied the entire Ethereum blockchain until last year's April. We basically downloaded all transactions, all smart contracts that have been there since 2015 to 2019 April. And we analyzed how many contracts are involved in most of the transaction.

How many contracts have most of the ether, how many contracts have most of the you know activities, and we find that there is a huge amount of inequality in the Ethereum smart contract. And we are doing this because bitcoin similarly bitcoin blockchain information has been similarly analyzed. And people found very similar patterns. And we wanted to see if Ethereum has the same.

And remember that the idea that the blockchain was put forth in the first white paper by Satoshi Nakamoto is that he wants to create a more just currency system and transactional freedom anonymity and all the government into lack of government interference and so on. But we find that the real social inequality that we are seeing in real economy is being reflected here almost entirely and probably even more.

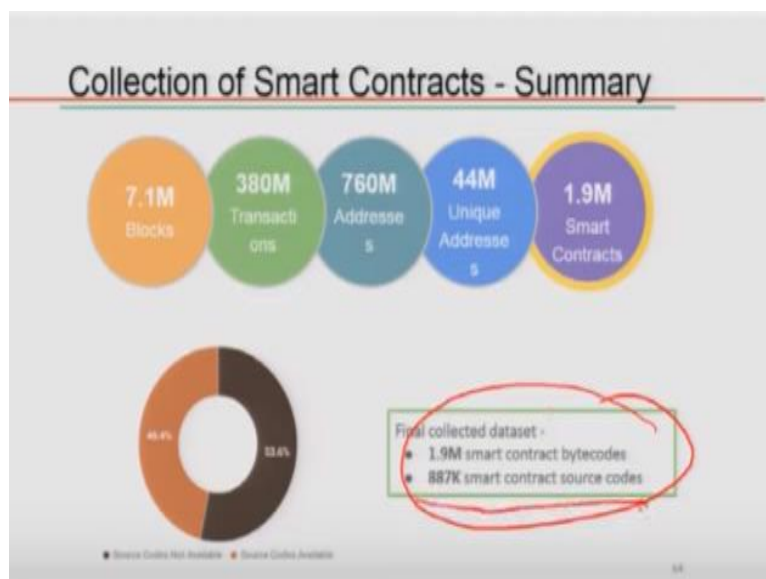
**(Refer Slide Time: 21:59)**

## Motivation

- DAO Attacks
- Ethereum is a public blockchain all the smart contracts are available
- To study the security trends and patterns in the real world smart contracts
- Analyse various smart contract verification tool on important contracts

So what I want to show you very quickly and over this after in the next session.

**(Refer Slide Time: 22:03)**



But I want to show you some of the results. So what we have found is, so until last year, we collected all the blocks. So there were 7 million blocks 7.1 million blocks at that time. There were almost 400 million transactions, about 760 million addresses, 44 million unique addresses among them. These are addresses that are referred in transactions. And then smart contracts there are we found 1.9 million smart contracts.

And out of which we found source code for about 53%. Because there is, you know some authors of the smart contracts also put their source code, whereas the other ones, they do not put their source code. So we did this. So we finally, we can collected 1.9 million smart contract byte codes, out of which half of which we can find. We could

find the source code. But for what I am going to tell you now does not require source code.

**(Refer Slide Time: 23:14)**

### Duplicity

- It is observed that out of 1.9M smart contracts, 1.8M (94.6%) are duplicates
- The most duplicated contract is User Wallet contract with over 650K instances
- However, only 2 out of the top 10 have verified source codes available
- It is observed that the top hundred contracts (0.1%) amount to a total of 1.72M (90.28%) occurrences in the dataset
- We call these High Occurrence Targets

Contract Bytecode Hash	src code	Contract Name/ Owner	Freq
2bf094df9076b24f2e6d8f1454f662	Y	User Wallet	651930
fa00c5b683d8f920ecc56452c1df224	N	?	158198
55f0329f9e6dbac461e933c66e0e2966	N	?	115132
dffc91bdac37abee7e8e9c82657f1f64	Y	Forwarder	99548
702adb219ba3238455a2a38c759798b	N	?	90489
92347eaf6e90eb27249343ca5c585945	N	?	78018
76d3bae3ec81aa704809a0912404ccaa	N	?	42868
42dbff3bccc3d1450068320ab64cd75	N	?	40456
1aa9943c89152c83cf788a5e74f4532	N	?	37534
125fb7c1a4489e08696034cf412a977	N	?	28255

So what I am going to show you is that see it is observed that out of 1.9 million smart contracts, 94.6 that is 95% are duplicates. So about 0.1 million that is about 100,000 or so smart contracts are actually original smart contracts. Most duplicated contract is of course, the user wallet contract, which has about 650,000 instances.

Only 2 out of top 10 contracts, which duplicated like when I say top 10, I mean, the ones which have been duplicated the most. So user wallet is the most duplicated and then the fourth duplicated is forwarded. This too we got the source code. Now observe that top 100 contracts, that is only 0.1% contracts out of the 100,000 original contracts, they actually get duplicated about 1.72 million.

So that means that 90% duplication are only about 100 contracts. So diversity of contracts is very low.

**(Refer Slide Time: 24:30)**



## Ether Balance

- The collected contracts contain a total of 10.88M ETH (worth US\$1.66B<sup>1</sup>)
- The most valuable contract is Wrapped Ether (WETH9). It contains roughly 2.4M ether (22%).
- It is observed that the top hundred contracts (0.1%) contain 10.7M (98.86%) ETH. This is worth US\$1.6B<sup>1</sup>
- We call these High Value Targets

Contract Bytecode Hash	ERC code	Contract Name/ Owner	ETH Balance
0e10248e905a82ff2070393ee0a2890c	Y	WETH9	2383501
e9b783c2301a1b2079d029a4f4814488	Y	Wallet	1430029
8026284c3fad07a5180297c0dfe09e	N	Genial's Cold Wallet	995099
108810e1a9180e325c08624f1a1c21e	N	Ethereum Foundation	646173
f5c40e048aca031a2eaf32ba04646e1	Y	MultiSigWalletWithDailyLimit	600341
c7010ef82217a9f2f0a0e82cf19907d	Y	MultiSigWalletWithDailyLimit	589093
4143a0500473d1647ae03423612a9e4	Y	Wallet	515036
c5fa4304689be1268719e04c1a4a689	Y	MultiSigWallet	396432
cdb19c392d0c440cd1121a0ca197ac	Y	MultiSigWallet	368023
1f086f1ad7ac2799011a048026bc436	N	?	292524

<sup>1</sup> as per exchange rate on 10<sup>th</sup> April, 2018

55

And then we have the wanted to see balances, right. So we found that top 100 contracts contain 99%, almost 99% of the Ethereum. So this is what we are worried about in real society today in the economy that like 0.1% of people actually control, let us say 70% of our country's assets and this is true in the US and other countries as well. So this is an inequality that is you know worrying economists.

And we would have thought that this, you know bitcoin and Ethereum these things will liberate us from such inequality of the society because, you know there is integrity and all that stuff. But actually, that is not happening. So only 100 contracts basically own 99% of the ether. So this is **very**, very worrying thing.

(Refer Slide Time: 25:26)

## Number of Transactions

- Smart contracts in our data-set were involved in 175M transactions (46% of all the transactions)
- 434K contracts (22.7%) had only one recorded transaction on the blockchain
- Most interacted with contract is the User Wallet
- We observe that 2500 contracts (2.5%) amount to a total of 90.37% (158M txns) of the total transactions done with smart contracts.
- We call these High Interaction Targets

Contract Bytecode Hash	ERC code	Contract Name/ Owner	Number of Txns
26f684d780f0b02472e6a88f1454f662	Y	UserWallet	583842
912778608c0976a28dc93fca4b91bb98	Y	EtherDelta	5273000
de379d8d0c4f52e45c99874e681bc70	Y	IBEX Exchange	3966000
1779d5757253c974734c46b1e98441e	Y	EttyCore (CryptoKitties)	3141000
17a3b0272ca30460eb26b894198f4c94	Y	BitToken (BB)	2955507
ee302cfa081c206484facb2c9543751	Y	TronToken (Tronix)	1906664
748f9df24379a8d14852844e99c46f4	Y	HumanStandardToken	1904722
c248f798c5ab00074907a54397cc6660	N	Polygon Exchange	1768964
fa00c8b4838bf920aee56852c1d2224	N	?	1681369
ca200836bc3f6fcb8bc9b42b3486c9	Y	Controller	1578993

56

And we saw that smart contracts on the on our data set were involved in 175 million transactions of which is 46% of all transactions. And we found that the 2500 contracts are responsible for 90% of the transactions. So which means that it is not hundred but it is still only 2500 contracts, which are doing most of the transaction so which means most of the accounts are excluded.

So I will go through this in even more detail in the next session. But what I want to caution you about is that this idea that blockchain and this pseudonymous blockchains will free up free society and all that stuff is not happening. And this is not only in Ethereum block, other people have done similar analysis. And they have seen similar trends in bitcoin blockchain.

So we will discuss this in more detail before we go into permission blockchain for the next several weeks. So but I think that this is something you should keep in mind when people advocate cryptocurrency, and you know cryptocurrency based society and all that, all that buzzwords. Okay, so we will see you next time.