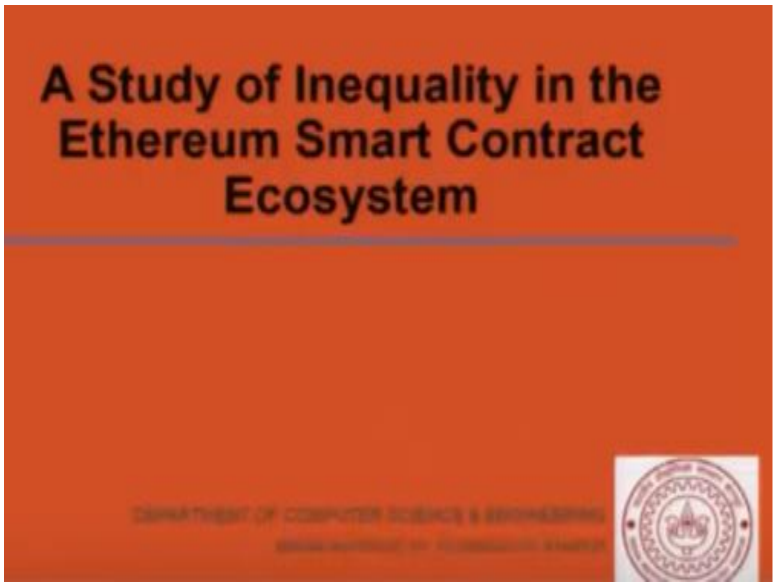
Introduction to Blockchain Technology & Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture - 17

Welcome to Blockchain Application and Technology course on NPTEL. So last time we spoke about Ethereum and gave you some idea about how to go about using Ethereum, pointed to the tutorial on the Ethereum.org website. And we started talking about the some of the downsides of the cryptocurrency blockchains such as Ethereum and bitcoin.

So today what I want to do is that we want to study some of the stuff that is quite concerning about cryptocurrency blockchains. And so that you have a very good idea about the risks and some of the pitfalls of this kind of permissionless cryptocurrency blockchains.

(Refer Slide Time: 01:14)

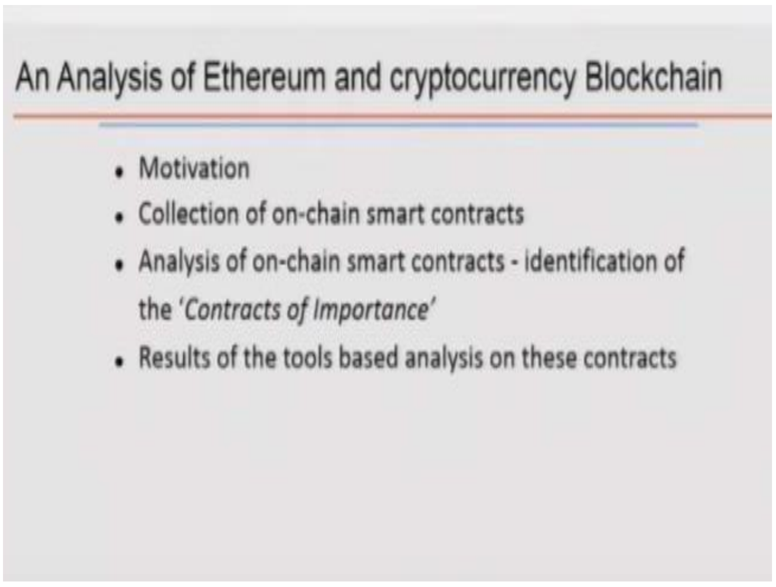


A Study of Inequality in the Ethereum Smart Contract Ecosystem

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

So the first thing we want to do is study the inequality in Ethereum smart contract ecosystem. And I kind of hinted at very quickly last time, how that works out, you know based on our own studies, and people have done similar analysis for bitcoin blockchain as well. So we will now go into a little more details about this inequality factor in Ethereum blockchain.

(Refer Slide Time: 01:42)



An Analysis of Ethereum and cryptocurrency Blockchain

- Motivation
- Collection of on-chain smart contracts
- Analysis of on-chain smart contracts - identification of the *'Contracts of Importance'*
- Results of the tools based analysis on these contracts

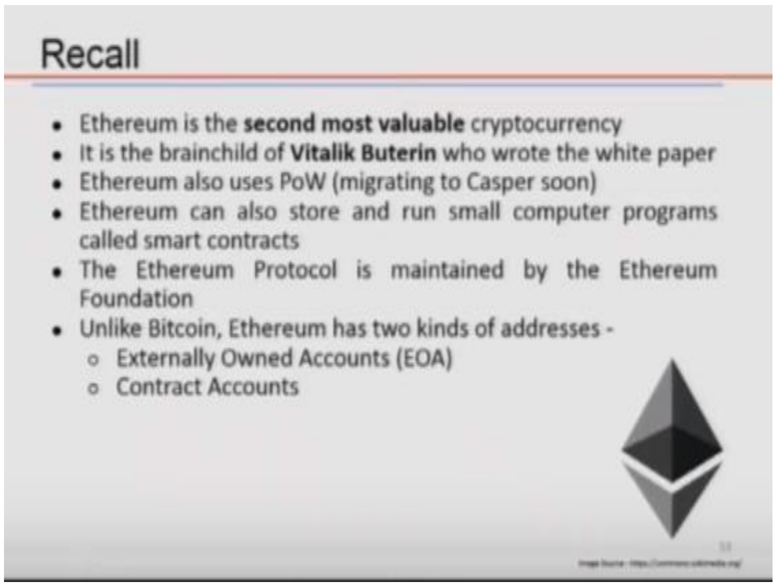
So what we want to do is that we want to first motivate why we are studying this. Second thing that we want to do is that, we want to tell you how we collected all the smart contracts that were available until last year April. And then we want to talk about how we analyze them and what kind of results we got from them.

So now regarding motivation, so there has been a lot of buzz about blockchain being an emancipator from the Crony capitalism, the Genesis block of the bitcoin blockchain back in 2008, by Satoshi Nakamoto actually quoted from the Times regarding the 2008 economic meltdown, because of the banking sector.

So the assumption that has been and lot of discussion and writings have come out that says that, well if you do, you know this distributed consensus based currency generation and use that for transactions and you can also maintain privacy of individuals then that is a very good way of getting rid of this whims and the various kinds of corruption that we see in the banking sector.

But turns out that it is not working out that way, right. So this is what we want to discover in the Ethereum blockchain. And that is why we did this study.

(Refer Slide Time: 03:24)



Recall

- Ethereum is the **second most valuable** cryptocurrency
- It is the brainchild of **Vitalik Buterin** who wrote the white paper
- Ethereum also uses PoW (migrating to Casper soon)
- Ethereum can also store and run small computer programs called smart contracts
- The Ethereum Protocol is maintained by the Ethereum Foundation
- Unlike Bitcoin, Ethereum has two kinds of addresses -
  - Externally Owned Accounts (EOA)
  - Contract Accounts

So but before we do that, let us again recall some of the important points about the Ethereum blockchain. So bitcoin is still the most valuable cryptocurrency and then a second most valuable cryptocurrency is Ethereum. It was kind of invented by Vitalik Buterin, a Canadian who wrote the first white paper which we discussed last time about the design, philosophy and how they thought of smart contracts.

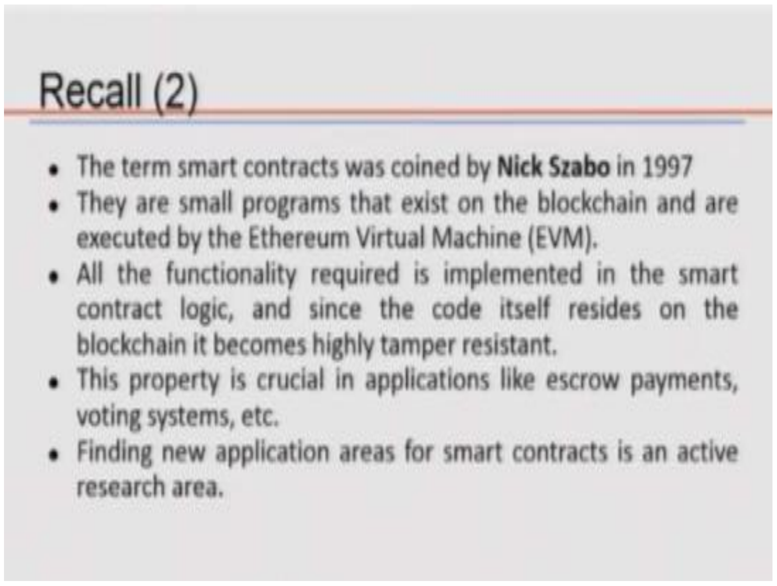
And how they actually wanted to overcome some of the cryptocurrency biases in the bitcoin blockchain and so on. And Ethereum also uses the proof of work, although they also have proof of authority, proof of stake and other possibilities. But until now it is mostly proof of work and the Mainnet is on proof of work. And the idea of Ethereum is that not only it is a cryptocurrency blockchain, but it actually executes distributed programs, which are basically run on the EVMs.

And these programs are universal programs, they are Turing-complete, and therefore all kinds of computation can be done, and therefore, the appeal of Ethereum in applications outside cryptocurrency is very high. And the Ethereum protocol is maintained by Ethereum Foundation. As we will discuss later, that Ethereum had a hard fork couple of in 2016.

And in the main branch that was there is called now Ethereum Classic and the fork branch is called Ethereum. And the reason for the hard fork was there was some extra rule that was that had to be enforced because of a particular type of cyber-attack on the original Ethereum blockchain, and we will talk a little bit about that later. And Ethereum as you know has two kinds of accounts.

One is accounts that are associated with real human participants, called externally owned accounts or EOAs. And then you have the accounts associated with smart contracts. And they are called contract accounts or CS.

(Refer Slide Time: 06:02)



Recall (2)

- The term smart contracts was coined by **Nick Szabo** in 1997
- They are small programs that exist on the blockchain and are executed by the Ethereum Virtual Machine (EVM).
- All the functionality required is implemented in the smart contract logic, and since the code itself resides on the blockchain it becomes highly tamper resistant.
- This property is crucial in applications like escrow payments, voting systems, etc.
- Finding new application areas for smart contracts is an active research area.

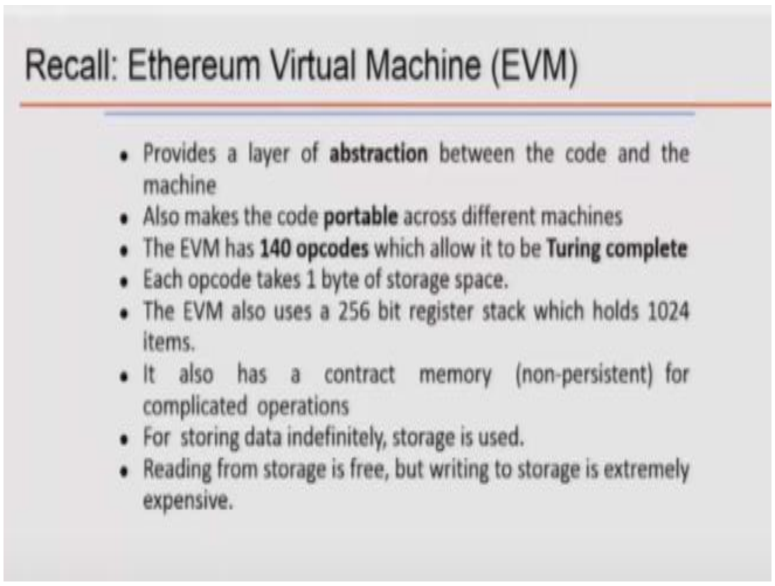
Also recall that smart contract is something that was new in Ethereum, because bitcoin scripting, language and bitcoin scripts are pretty limited in what they can do. And these are smart contracts are Turing-complete programs, which are recorded on the blockchain, and they are executed on the Ethereum virtual machine. And every node eventually has to participating and maintaining the blockchain has to eventually execute the programs on their local EVM.

And the consensus mechanism works so that once a block is mined, then all the steps that were executed for all the smart contracts in the block that has been accepted and percolate in every node that is maintaining the blockchain has to also execute those. And the functionality is implemented in the smart contract logic. And the code itself resides on the blockchain and hence you cannot tamper with it, which actually is a good thing and a bad thing.

The good thing is that the programs are tamper resistant. The bad thing is that if you find a bug, then you cannot just go and fix the bug. And there are other mechanisms to stop the contract and then recreate an completely new contract when it comes to that. The smart contracts are very critical to implementing things like escrow payments, voting systems, student registration system, all kinds of applications that are not really cryptocurrency applications.

Although for execution of smart contracts, you have to pay in gas and gas costs ether, so you need to have Ethereum balance on your account in order to make transactions on those smart contracts. And finding new application areas for smart contracts is an active research area. So towards the end of this course, we will talk about some of the applications.

(Refer Slide Time: 08:08)



Recall: Ethereum Virtual Machine (EVM)

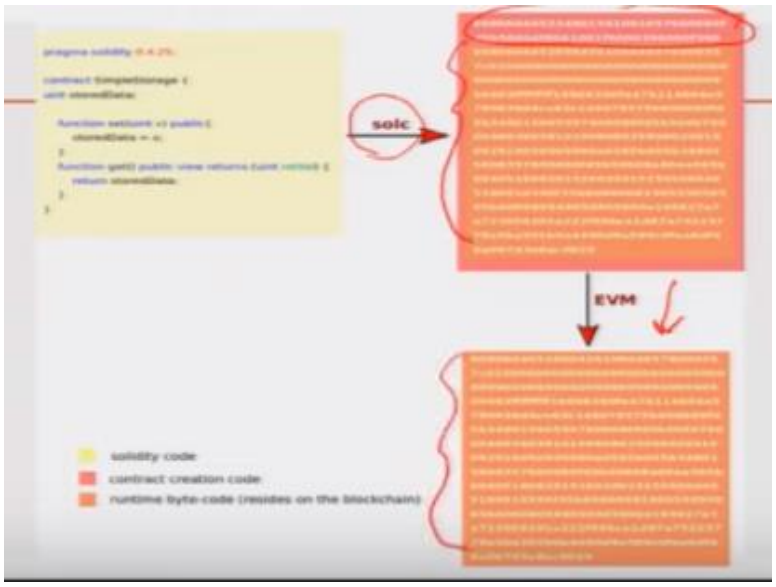
- Provides a layer of **abstraction** between the code and the machine
- Also makes the code **portable** across different machines
- The EVM has **140 opcodes** which allow it to be **Turing complete**
- Each opcode takes 1 byte of storage space.
- The EVM also uses a 256 bit register stack which holds 1024 items.
- It also has a contract memory (non-persistent) for complicated operations
- For storing data indefinitely, storage is used.
- Reading from storage is free, but writing to storage is extremely expensive.

Ethereum virtual machine, if you remember is a layer of abstraction between the code and the machine. And also it makes the code portable across different machines. It has about 140 opcodes. And these opcodes are designed so that it is Turing-complete and as we discussed in the design principle of Ethereum, these opcodes are pretty low level actions.

So any high level action that needs to be taken will be a combination of sequence proper sequencing of this low level opcodes to obtain a high level functionality. And each opcode is 1 byte long. And as we know that EVM has a stack, registered stack based execution model with 1024 items, and each of the register is 256 bit long.

And it has contract memory, which is non-persistent, which is which does not go on the blockchain and then it has the storage and the storage is the permanent state of the smart contract. And therefore, they go on the blockchain. And when the blockchain wants to read the current state that is free, but if you want to write the current state, then you have to pay in gas.

(Refer Slide Time: 09:25)



```
pragma solidity ^0.4.25;

contract SimpleStorage {
uint storedData;

    function set(uint x) public {
        storedData = x;
    }
    function get() public view returns (uint retVal) {
        return storedData;
    }
}
```

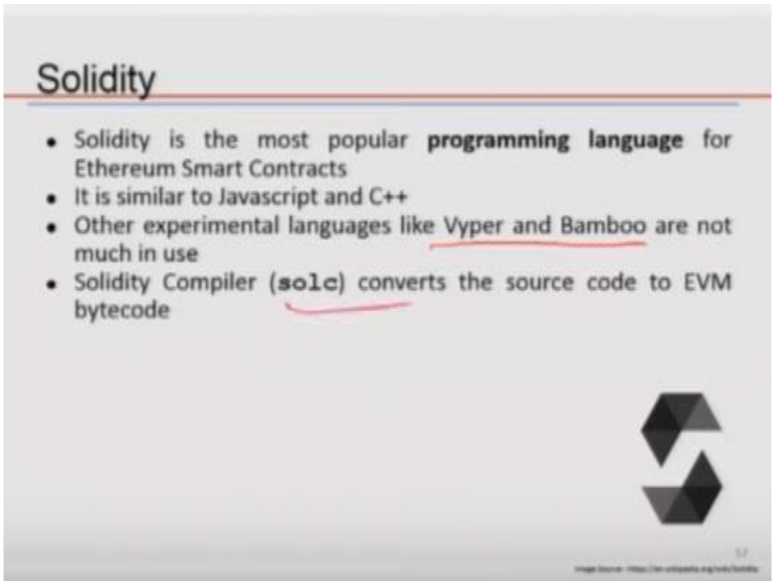
🟨 solidity code

🟥 contract creation code

🟧 runtime byte-code (resides on the blockchain)

So as we know that, the source code is written in language like Solidity, and the Solidity is compiled into the bytecode. And that bytecode contains the smart contract creation code and the actual code that will be translation of the various functions in the contract and its storage information. So when it goes when it gets deployed, then the entire bytecode goes into the blockchain and so the part that is the actual contract code in bytecode form then gets persisted into the blockchain.

(Refer Slide Time: 10:06)

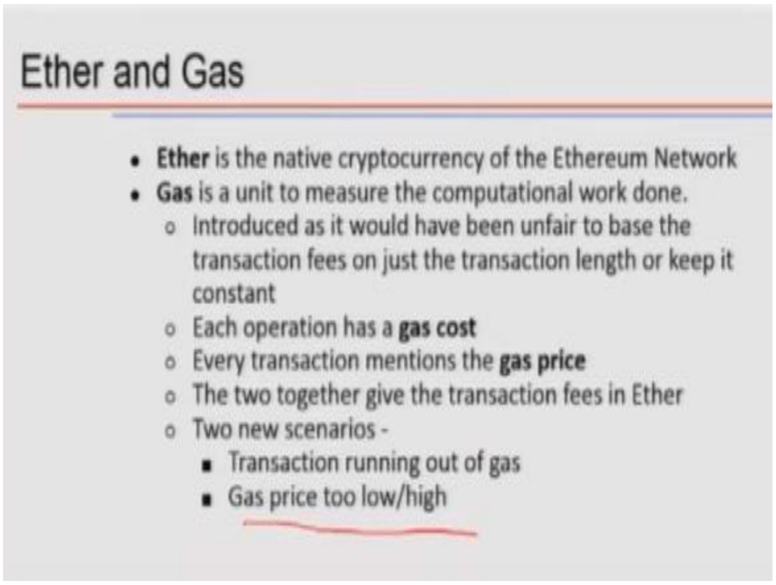


Solidity

- Solidity is the most popular **programming language** for Ethereum Smart Contracts
- It is similar to Javascript and C++
- Other experimental languages like Vyper and Bamboo are not much in use
- Solidity Compiler (`solc`) converts the source code to EVM bytecode

Now Solidity as we know is a programming language for smart contracts. We talked about this cursorily before. It is similar to JavaScript and C++. There are some other experimental languages that have come, but they are not so popular and solidity compiler converts the source code to the EVM bytecode. So all that stuff we have already covered.

(Refer Slide Time: 10:30)



Ether and Gas

- **Ether** is the native cryptocurrency of the Ethereum Network
- **Gas** is a unit to measure the computational work done.
  - Introduced as it would have been unfair to base the transaction fees on just the transaction length or keep it constant
  - Each operation has a **gas cost**
  - Every transaction mentions the **gas price**
  - The two together give the transaction fees in Ether
  - Two new scenarios -
    - Transaction running out of gas
    - Gas price too low/high

Ether is the native cryptocurrency. Gas is a unit to measure the computational work done. So the transaction fees are basically based on computational requirements that the transaction imposes on the nodes and not just the transaction length and each operation has a fixed gas cost and then every transaction mentions the gas price that the invoker of the transaction wants to pay and these two together give the transaction fees in the ether.

And then there are two scenarios, one is transactions may run out of gas or if it has a very large number of steps or if it has a loop that does not terminate properly, and then the gas price may be too low or too high. And if it is too low, then it may not be any minor may not pick up that particular transaction. And if it is too high, then there are obviously somebody is trying to prioritize and making unfair taking unfair advantage plus possibly to get their transaction into the next block.

(Refer Slide Time: 11:46)

Collection of On-Chain Smart Contracts

So the question now is that we want to analyze what you see it is by looking at the history of all blocks and transactions and all the smart contracts that we have seen until a certain date which in our case would be sometime in April last year. All the blocks and the all the transactions and the smart contracts we collected and put in into local storage so that we can analyze them.

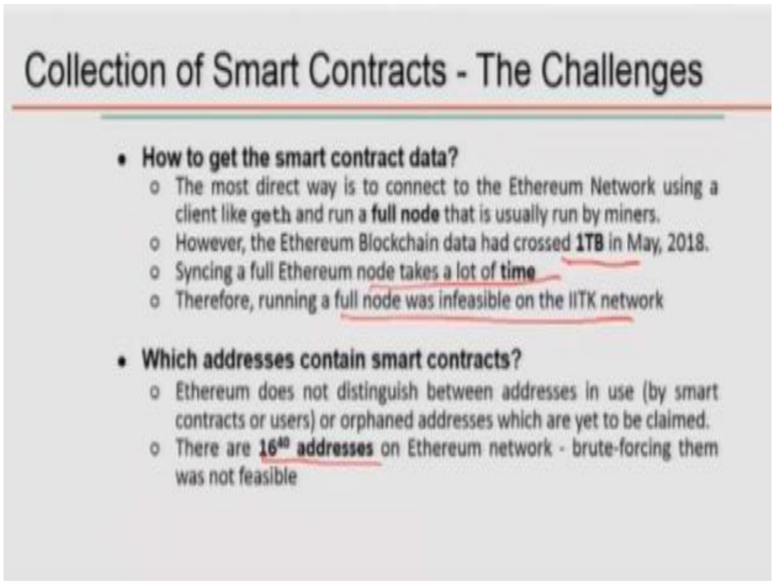
And this analysis was made with two objectives in mind. One is that is the hypothesis that the this permission less cryptocurrency blockchains provide more democratization of market and more democratization of the operations that happens with respect to transaction purchasing and all that stuff or even storage of value.

And second thing we want to analyze was the there has been a lot of cyber-attacks on the Ethereum blockchain and other blockchains as well, including bitcoin, but in Ethereum, because of smart contracts, which are programs, so there are bugs in such programs. And because of those bugs, people exploit the bugs to actually launch various kinds of attacks to actually steal money. And we will see some of those cases.

And so we wanted to understand what are the different kinds of bugs, which are the common bugs, and how we can actually get rid of them. There are a lot of tools in the market. Which tools work better when you write a smart contract? You might test it on the testnet for a while. But testing as we know program testing is not a foolproof method for verification.

So there are tools that actually take the source code or the bytecode of the contract, and then analyzes them using various techniques and try to figure out if there are bugs which can be exploited. So we want to do both. So we collected all this smart contracts that were on chain until the time that we stopped collecting them.

(Refer Slide Time: 14:11)



Collection of Smart Contracts - The Challenges

- **How to get the smart contract data?**
  - The most direct way is to connect to the Ethereum Network using a client like `geth` and run a **full node** that is usually run by miners.
  - However, the Ethereum Blockchain data had crossed **1TB** in May, 2018.
  - Syncing a full Ethereum node takes a lot of **time**
  - Therefore, running a full node was infeasible on the IITK network
- **Which addresses contain smart contracts?**
  - Ethereum does not distinguish between addresses in use (by smart contracts or users) or orphaned addresses which are yet to be claimed.
  - There are **16^{40} addresses** on Ethereum network - brute-forcing them was not feasible

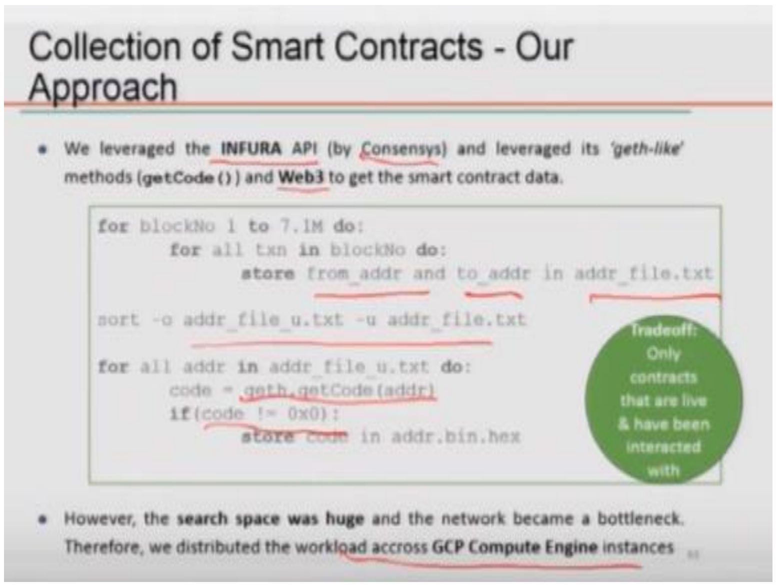
So if you run a full node, so as we know in bitcoin also we discussed this that if you run a full node, that means you participate, you keep the entire blockchain. Every time a new block is created, you are supposed to validate it, and then connect it to the blockchain and all that activities. And if you are a lightweight node, then you may just only keep the block headers and so on in order to just verify transactions.

So if you want to run a full node, then you have to download locally, the entire blockchain and if you do that, then I have a local copy of all the blocks and transactions and the smart contracts. And I can do analysis. But Ethereum blockchain data had crossed over 1 terabyte back in May 2018. And therefore, syncing a full Ethereum node takes a lot of time. Therefore running a full node was infeasible on our local network.

So what we did is that we wanted to only collect the smart contracts and wanted to analyze their behavior. So now the question is, there are many addresses. In fact, ideally, there are 16 to the 40 addresses on the Ethereum network. And therefore, how do I know which ones are associated with smart contracts and which ones are not? So I have to somehow figure out that which addresses actually have smart contracts on them.

Or I have to do some other way to actually figure out how to collect all the smart contracts associated with the addresses that are that correspond to smart contracts.

(Refer Slide Time: 16:02)



Collection of Smart Contracts - Our Approach

- We leveraged the **INFURA API** (by Consensys) and leveraged its *'geth-like'* methods (**getCode()**) and **Web3** to get the smart contract data.

```
for blockNo 1 to 7.1M do:
        for all txn in blockNo do:
                store from_addr and to_addr in addr_file.txt

sort -o addr_file_u.txt -u addr_file.txt

for all addr in addr_file_u.txt do:
        code = geth.getCode(addr)
        if(code != 0x0):
                store code in addr.bin.hex
```

Tradeoff: Only contracts that are live & have been interacted with

- However, the **search space was huge** and the network became a bottleneck. Therefore, we distributed the workload accross **GCP Compute Engine** instances

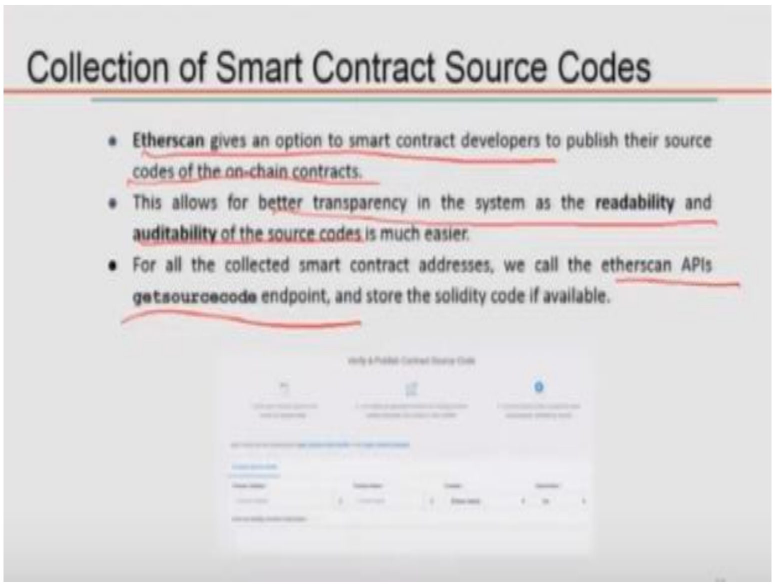
So we use the something an API called INFURA API by consensus, and its method, gate code and a Web3 interface to get the smart contract data. So what we did is that we first went through all the blocks, and for all transaction, we got the from address and to address into a file, then we sorted them. And then we basically remove the redundancy. And then we basically called get code.

And if there is a code associated with that address, then we got the code, right. So that is what we did. So basic idea is that you collect all the possible addresses that are currently either appearing as a to address and or a from address. And then if there is associated and then go through all the addresses, ask whether it has code. If it has code associated with it, then it is a smart contract code and we downloaded them.

So we had to actually use Google Cloud Platform Compute Engine to do this, because our network was not able to do this fast enough, right. So that is the way we did it. You can also do it if you are interested in this kind of analysis, because we did the analysis only until last April or so.

And there is lot more things have happened since last one year and you might see different patterns or at least some correction in the trends that we are going to report here. So you are welcome to do this kind of an experiment.

(Refer Slide Time: 17:44)



Collection of Smart Contract Source Codes

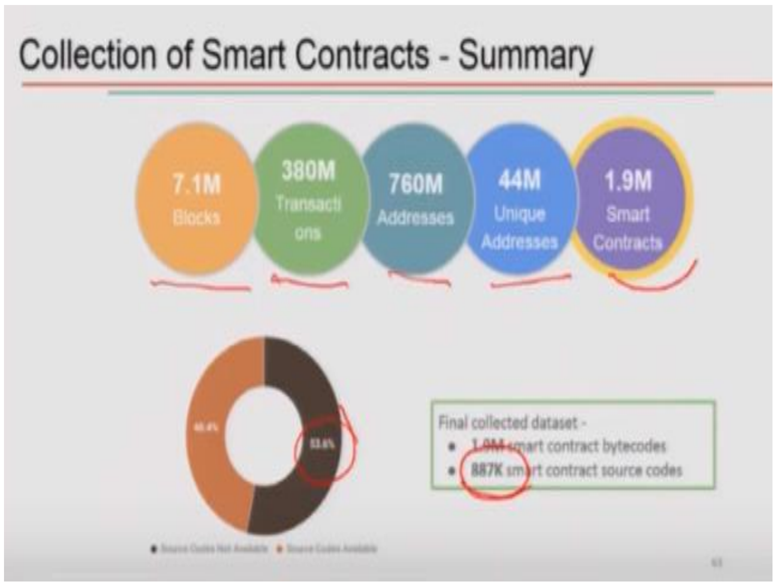
- **Etherscan** gives an option to smart contract developers to publish their source codes of the on-chain contracts.
- This allows for better transparency in the system as the **readability** and **auditability** of the source codes is much easier.
- For all the collected smart contract addresses, we call the etherscan APIs **getsourcecode** endpoint, and store the solidity code if available.

Now for some of the tools that we are going to use for analysis of the bugs. We also need the source code because not all tools work on the bytecode directly if someone requires the source code. So the question is the what we downloaded like this, the process that I just described, was actually giving me bytecodes. However, Etherscan allows smart contract developers to publish their source code of all the unchained contracts.

So when you deploy a smart contract, you only deploy the bytecode. But you can also go to Etherscan, and actually upload the corresponding source code if you want. So this gives better transparency, readability and auditability of source code. So for all collected smart contract addresses, we call the Etherscan API to get source code and stored the Solidity code if available.

So not all bytecode has corresponding source code. But whichever has this source code, we collected them. So let us see what we got.

(Refer Slide Time: 18:52)



Collection of Smart Contracts - Summary

Final collected dataset -
- 1.9M smart contract bytecodes
- 887K smart contract source codes

So we got at that time, so at the time when we stopped, I do not remember the exact date, we had 7.1 million blocks, 380 million transactions, 760 million addresses, 44 million unique addresses because of to and fro will have overlap. And then we found 1.9 million smart contracts. That is about 2 million smart contracts that are currently at that time was on chain of the Ethereum blockchain.

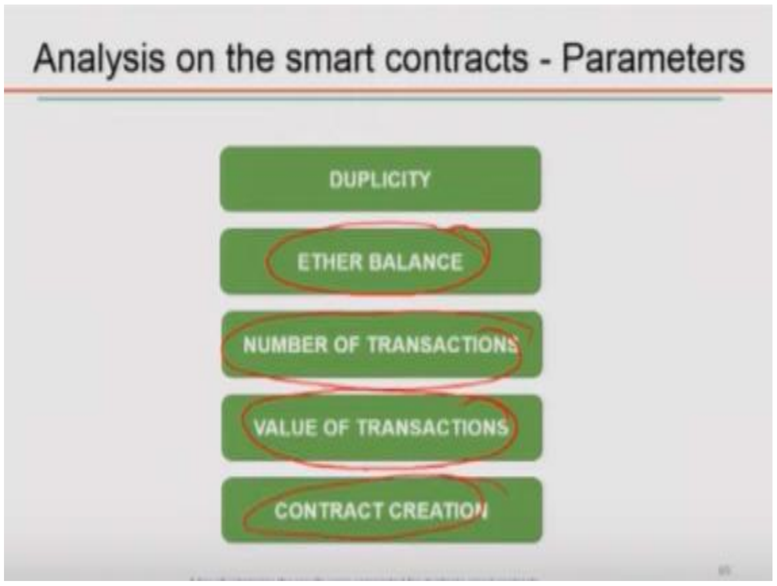
And source code available was 53.6% cases we found the source code. So the number of contracts for which we have the source code is about 887,000. That is about 890,000 or 900,000 so to speak out of 1.9 billion smart contracts. So that is the situation. That is our base data on which we do the analysis.

(Refer Slide Time: 19:56)

Analysis of On-Chain Smart Contracts

So how do you do that? What kind of analysis do you want to do to bust the myth that this blockchain cryptocurrency is highly democratizing?

(Refer Slide Time: 20:13)

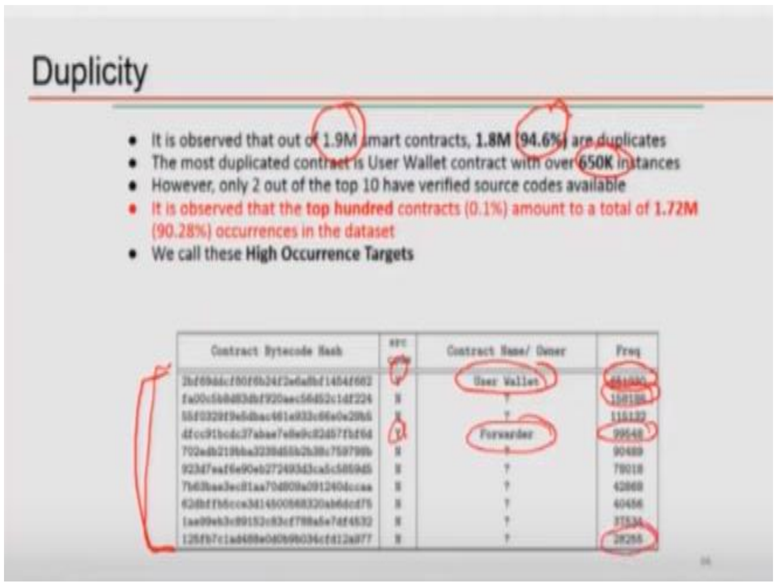


Analysis on the smart contracts - Parameters

- DUPLICITY
- ETHER BALANCE
- NUMBER OF TRANSACTIONS
- VALUE OF TRANSACTIONS
- CONTRACT CREATION

And so what we wanted to check is which of the contracts have more ether balance? Which of the transactions are doing most of the transactions? What are the values of the transactions and then the contract creation and duplicity of smart contracts. So these are the some of the analysis that we have done.

(Refer Slide Time: 20:33)



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
2bf69ddcf00f6b24f2e6a8bf1454f602	✓	User Wallet	[illegible]
fa00c5b8d83dbf920aec56452c14f224	✗	?	158138
55f0329f9e5dbac461e933c66e0e29b5	✗	?	116132
dfcc91bcdc37abae7e8e9c82457fbf64	✓	Forwarder	99548
702edb219bba3238455b2b38c759798b	✗	?	90489
923d7eaf6e90eb27249343ca5c5859d5	✗	?	78018
7b63bae3ec81aa70d809a091240dccaa	✗	?	42868
62dbffb6cce341450088320ab6dcd75	✗	?	40456
1ae99eb3c89152c83cf788a5e7df4532	✗	?	37534
125fb7c1ad488e0d0b9b034cf412a877	✗	?	28255

So let us look at duplicity. Duplicity means that you write a smart contract, and everybody wants their own smart contract, which requires the same functionality. So they directly take your smart contract and duplicate it. For example, cryptocurrency wallet, user wallet is something that everybody needs every user externally owned account needs in order to keep their ether balance, and also their private key and so on.

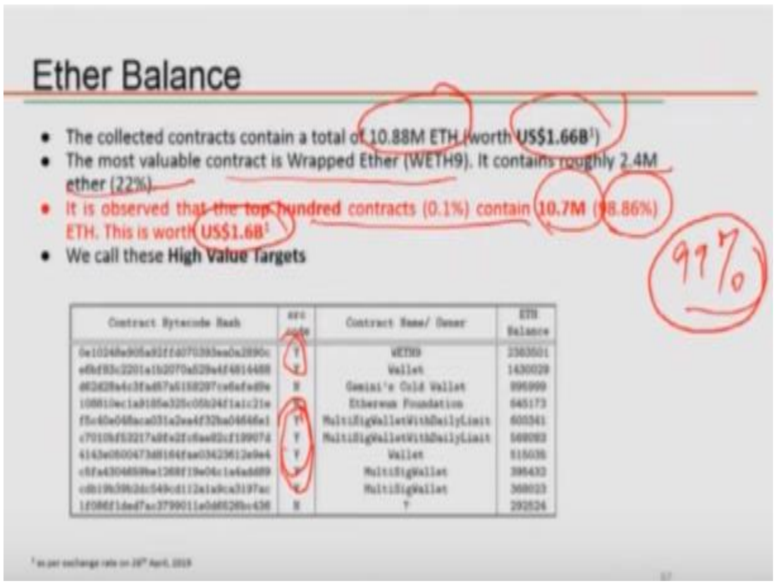
So therefore, user wallet contract has been the most duplicated some smart contract with 650,000 instances. So remember out of 1.9 million smart contract, we found that 94.64% are duplicates. So which means 1.8 million of these smart contracts that are on chain are actually duplicates of other contracts. So which means there are only about 100,000 unique contracts that we could figure out.

And then the most commonly duplicated one was the user wallet. So here is a list of the bytecode hash for the top 10 duplicated contracts. And on the top, you see the first one that is most duplicated was 651,000 times about 652,000 times. And that is the user wallet. How do we know it is user wallet because they also have the source code among the source code that we could download.

And then the fourth one forwarded, which also has a duplication of about 100,000 times and that one is also the source code is available. For the rest top 10, which has been duplicated so many times are not, source code is not available. So we do not know the contract name or the owner. But the second one highest one has been duplicated about 158,000 times and the third one 115,000 times and the 10th one is about 28,000 times.

So you see that most of the 90 almost 95% of the contracts on chain are duplicate. So we have to now focus on the 100,000 contracts which are unique to understand what they do.

(Refer Slide Time: 23:08)



Ether Balance

- The collected contracts contain a total of 10.88M ETH (worth **US$1.66B**[1])
- 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**[1]
- We call these **High Value Targets**

99%

Contract Bytecode Hash	src code	Contract Name/ Owner	ETH Balance
0e10248e905a92ff4070393ea0a2890c	Y	WETH9	2363501
e6bf83c2201a1b2070a529a4f4814488	Y	Wallet	1430029
d62d28a4c3fad67a6158297ce6afed9e	N	Gemini's Cold Wallet	895999
108810ec1a9185e325c05b24f1a1c21e	N	Ethereum Foundation	645173
f5c40e048aca031a2ea4f32ba04646e1	Y	MultiSigWalletWithDailyLimit	600341
c7010bf53217a8fe2fc6ae82cf19907d	Y	MultiSigWalletWithDailyLimit	569083
4143e0800473d8164fae03423612e0e4	Y	Wallet	515035
c6fa4304659be1268f19e04c1a4add89	Y	MultiSigWallet	398432
cdb19b39b2dc549cd112a1a9ca3197ac	Y	MultiSigWallet	368023
1f086f1ded7ac3799011e0d6528bc436	N	?	292524

[1] as per exchange rate on 28th April, 2019

So we then for all the collected contracts, we found how much ether balance they contain. That is the power of the particular contract is how much balance they have. For example, if the smart contract is basically the smart contract that is running an exchange for Ethereum then they will have a pretty large amount of ether stored in them.

But if it is you and I user, we might have even a fraction ether right. So the question is, what is the situation with respect to that? So we found that time the total Ethereum that we found in the contracts as balance is about $1.66 billion worth US dollar worth. And most valuable one was Wrapped ether and it had 2.4 million ether, which is 22% of the total ether across all the contracts.

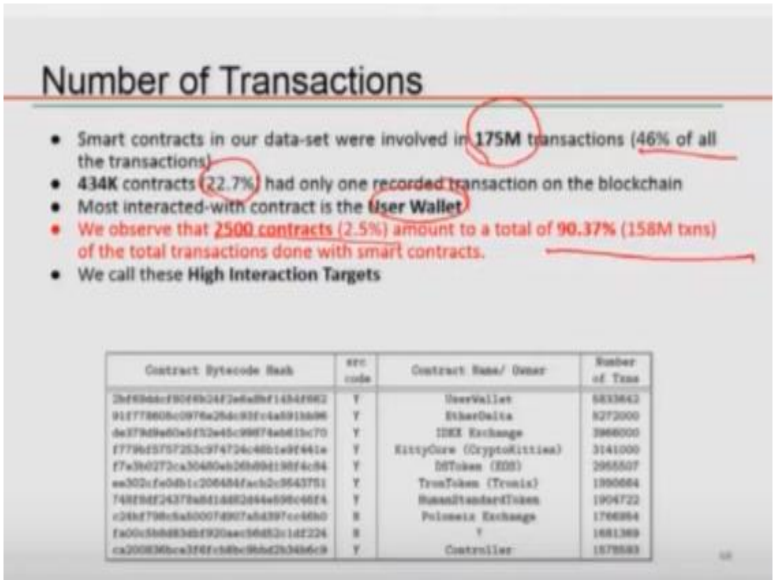
It is also observed that the top 100 contracts that is only 0.1% contain about 10.7 million that is 98.86. That is 99% of the money was with only 100 contracts. And that was worth about 1.6 billion. So these are called high value targets because if a cyberattacker wants to target a particular contract, and if that contract has bugs, then that will certainly be tried to be breached in order to make money.

And that has happened, right. So we will talk about that later. But so we found that the top 10 cases we found 1, 2, 3, 4, 5, 6, 7 seven of them we found the source code also. So and then other ones, we do not have the source code. But most of them are some kind of a wallet. See multisig wallets is when you have a wallet that requests a multiple people's signature or two out of three or three out of five kind of signature.

So usually they are not individual accounts or they are accounts owned by a corporation or some kind of an entity that is a multisig wallet. So **so** naturally they have more balance. But this is kind of scary that the thing that blockchain that is supposed to be very, you know democratizing, you know technology actually has all the all its worth concentrated with only about about 0.1% of the contracts.

Now remember the even the accounts that are maintained that are users accounts that is externally owned accounts they also have contracts because their wallet is a contract right? So that means that there are individual or individuals behind those contracts which have all this ether. So that is the situation here.

(Refer Slide Time: 26:23)



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	src code	Contract Name/ Owner	Number of Txns
[illegible]	Y	UserWallet	5833642
[illegible]	Y	EtherDelta	8272000
[illegible]	Y	IDEX Exchange	3966000
[illegible]	Y	KittyCore (CryptoKitties)	3141000
[illegible]	Y	DSToken (EOS)	2955507
[illegible]	Y	TronToken (Tronix)	1990684
[illegible]	Y	HumanStandardToken	1904722
[illegible]	N	Poloniex Exchange	1766864
[illegible]	N	?	1681369
[illegible]	Y	Controller	1578583

And the number of transactions, we saw that smart contracts are involved in 175 million transactions which is 46% of all the transactions. Out of, because transactions can be done by contracts or can be done by clients. So about 22.7% had only one recorded transaction. So they are basically not very participating. That is 22.7% of them are not very participatory in the blockchain. Most interacted with contract is user wallet.

And we found that 2500 contracts that is only 2.5% amount to a total of 90.37 that is almost 91% of the total transaction. So again, we see a concentration of wealth and we see concentration of participation among this very small number of participants in the network. So when we come back, we will talk about the other aspects of this analysis. And then we will go into the issue of security of the Ethereum blockchain.