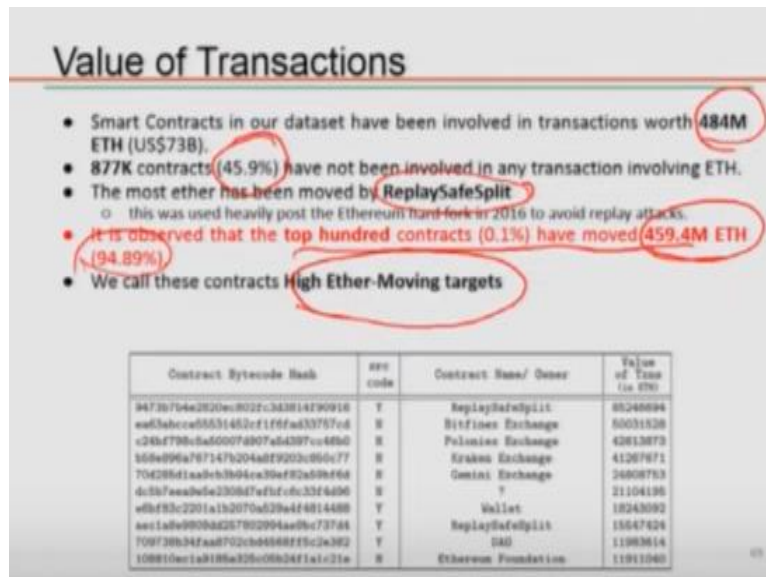


**Introduction to Blockchain Technology & Applications**  
**Prof. Sandeep Shukla**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology-Kanpur**

**Lecture - 18**

Welcome back. So we have been talking about the issue of inequality in Ethereum smart contracts.

**(Refer Slide Time: 00:25)**



And we already saw that the total balance is very much concentrated with only a very few smart contracts about 0.1%. And then we also saw that the participation the transactions are done mostly by top 2.5% of the smart contracts which means that even participation is very low from the most of the nodes of the network. Now value of transactions, right.

So smart contracts are involved in about 484 million ether which is about 73 billion US dollar worth of transactions. Now out of which about 877,000 that is about 900,000 contracts have not been involved in any transaction involving ether. So that is 45.9% of the network is quiet, they are not doing any transaction. And or they are doing transactions which are but not in money type of transaction.

Now most ether was moved by ReplaySafeSplit. And this happened when there was a hard fork in 2016 after the DAO attack, and we will talk a little bit about that later. So therefore, there was a split and it was a hard fork and the earlier branch became the

Ethereum Classic and the new branch is the Ethereum. And that put some extra restriction.

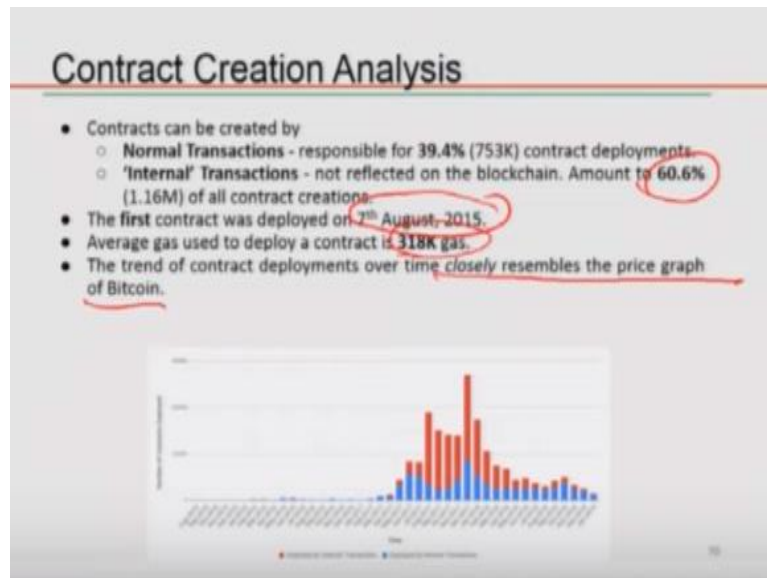
As we know that hard forking happen when you put extra restrictions on the transactions, and therefore the nodes that do not catch up will not ever get their blocks or transactions and blocks accepted. So that is how the hard fork happen. And obviously, the lot of ether was now sent to the new branch. And that is what the split was about.

So top 100 contracts we found have moved out of this 484 million ether transactions. Almost 460 million that is about 95% of the transactions, the money movement was done by about 100 contracts out of 1.9 million smart contracts on chain of which obviously 100,000 are the only the non-duplicate ones. So out of which only 100 actually have moved most of the 95% of the ether.

So this is these are called high ether moving targets. These could also be target of cyber-attack because obviously, if they are moving that much ether, their balance is high. And also when they make this high moving transactions, if there is a bug through which they attack, that transaction can be rerouted to an attacker's account or something, they will obviously try that.

They will look over the code to see if there are some vulnerabilities or bugs that they can exploit.

**(Refer Slide Time: 03:26)**



So contract creation analysis. So contracts are created by either normal transactions or by internal transactions. Normal transactions can be found on the blockchain in the blocks. But internal transactions, internal transactions are transactions, usually between smart contract to smart contract. And internal transactions are not recorded on the blockchain. But they amount to about 60.6% of all contract creation.

The first contract was deployed in August 2015 which is about 6, 7 years after the bitcoin blockchain right? So it is a new kitten around the block. And then turns out that it has easily come up to the second position because of the all these different feature that Vitalik Buterin thought about and so on. The average gas used to deploy a contract was about 318,000 gas units.

And the trend of contract deployments over time closely resembles the price graph of bitcoin. And this is interesting, because as the bitcoin prices went up and up, overall cryptocurrency got a boost, right? And then more and more people started getting into the cryptocurrency creating applications on the blockchain for various kinds of things.

For example, creating lotteries, creating Ponzi schemes, creating pyramid schemes as well as creating DAOs or distributed you know applications, that would actually be like a, like a venture capital type of functionality. All those things started coming, and they actually closely resembled the bitcoin price graph.

**(Refer Slide Time: 05:29)**

## Contract Creation Analysis

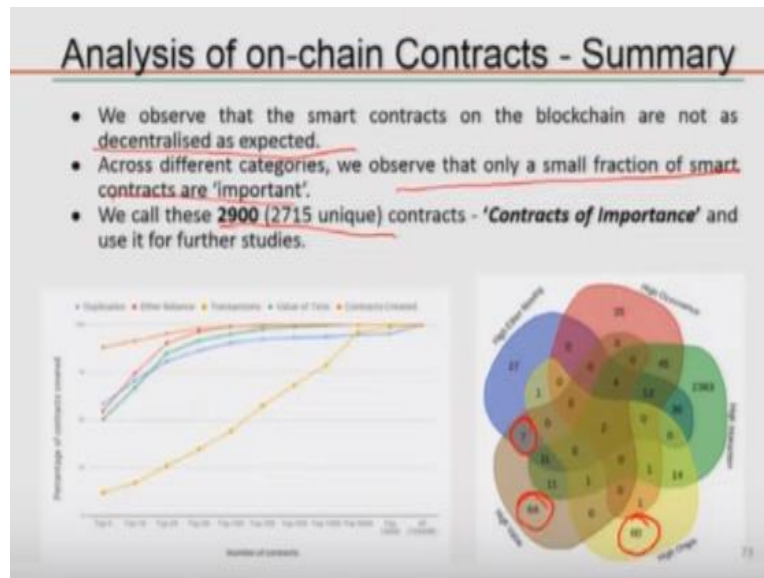
- For the 1.16M contracts deployed using internal transactions, it is observed that these contracts have been deployed by only 9228 unique contract addresses. This number reduces to 2420 contracts on considering duplicates.
- It is observed that the top 100 (0.1%) contracts have deployed 1.14M contracts (98.93%).
- We call these High Origin Targets.

Contract Bytecode Hash	Total Contracts Deployed	Unique Contracts Deployed
b071cebdaac85e6cfa20b94e78314386	651930	1
23eed1c018699f2aa9217e487851262b	115132	1
07459966443977122e639cbf7504c446	99547	1
1409c3e3b085b70674d7446f3d30df36	90489	1
3664d689499a9ec1538b9443c57bbc0	62313	2
1093df22a683b4efffcb608adcf6689	11369	1
11f7b022128ecfca93375856a2613c	7904	1
06b915a788451c8a5f44715914d8ca5c	7054	1
24382304429430b1706f4089d196d265	6693	1
1a5383732e90b3fdbf70413d490b2684	6144	1

So 753,000 contracts deployed by normal transactions have been deployed by only about 57,600 accounts. So that is basically about, I would say about 12, 13 times less, which means that about 1213 contracts were deployed by each of these 57,600 accounts on an average. So 1.16 million contracts deployed through internal transactions. These contracts have been deployed by only about 9000 to 28 unique contract addresses.

And this number reduces to 2420 if you only consider unique contracts. So top 100 contracts have deployed 99% of these contracts. So when you leave it to these programs rather than human beings deploying new contracts, it seems that it is more lopsided, and only about 100 smart contracts have deployed over 1 million contracts, which is about 99% of all the contracts that were created by internal transaction and these are called high origin targets because they are very active, right. So about 100 of them are very active.

**(Refer Slide Time: 07:01)**



So in summary, we observe that the smart contracts and blockchain are not as decentralized as expected in a social sense. And across different categories, we observe that a small fraction of smart contracts are important. And we call this and we identify this and there are about 2715 unique such contracts and we call them contracts of importance, and we use it for further studies.

Now the other question that we have is that, what is this that are these things about this 2715 unique contracts, how much they overlap? For example, high ether moving contract, do they overlap also with high occurrence that is high duplication contracts or high value contracts. So we see that you know about this high value contracts has quite a bit of overlap with the high ether moving contracts.

Similarly, high occurrence contracts has quite a bit of overlap with the high ether moving contracts, as well as with high interaction contract and little less overlap with the high origin contracts. But, among the high value contracts, we found that 64 are not either high ether moving, or high occurrence or high interaction or high origin. 80 of the contracts are only high origin.

But they do not either have high value or they do not have high interaction. Similarly, 7 contracts are both high ether moving and high value and so on. So that gives us the 100 that we saw in terms of high ether balance, high ether moving. This is slightly better in the sense that at least 2900 or so contracts are somehow coming up as either high occurrence or high interaction, high origin, high value and so on.

So it is about 2.5% of the total participant demography about 2.9% and so on. 2.7% of the unique contracts are somehow contracts of importance. So which means about 97.3% of the contracts are totally not so important. So which basically is reflects a sort of a situation where most of the contracts and most of the participants etc., are not really participating.

And therefore the myth that this cryptocurrency blockchains are somehow equalizing technological force is not so much validated by this data.

**(Refer Slide Time: 10:05)**



So now we move to the next issue that is security of Ethereum blockchain. Now there is a myth and I have seen this amongst students as well as IT professionals. They think that blockchain means secure. Now blockchain that when you first of all, when you design and implement the blockchain correctly with proper cryptography, strong cryptography and with strong hashing algorithms and so on.

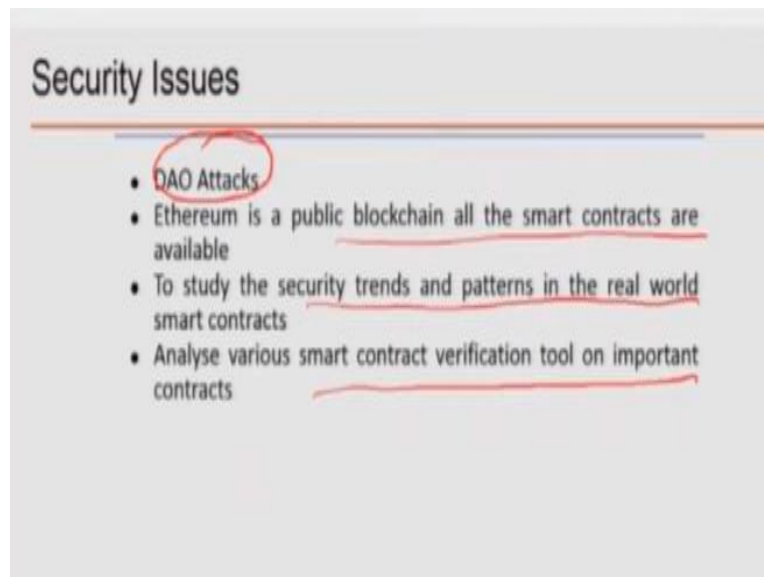
Not so you know conflict resistant, hash functions and cryptography, proper cryptography and so on, then I can say that the data that goes into the blockchain cannot be easily tampered. So it is kind of tamper resistant. So which gives me data integrity. But security is confidentiality, security is availability and security is integrating. Now none of these things even data integrity is after the fact right.

So the data that has been has had a consensus on cannot be tampered with, right. But if a smart contract is invoked by an attacker in the wrong way, and ethers have been moved, because there was a bug in the smart contract, then that transaction is a cyber-attack is a result of a cyber-attack. But it when it will get recorded in the blockchain and in the future, it will stay there in the blockchain unless you do something else to correct course, which is an issue that was that came up in 2016.

But security is not given in blockchain. Security is something that is dependent on a lot of issues, including the correctness of smart contracts, whether it has vulnerability and security bugs, it has issues with the design of the protocol, it has issues with the design of the architecture, it has to do with cryptography that is chosen, it is to do with the ability to have the consensus algorithm to be either, you know highly, you know highly correct in the sense that it is not compromised.

All this stuff has to do with security. So people have a very wrong notion that security is given in the blockchain, but that is not the case.

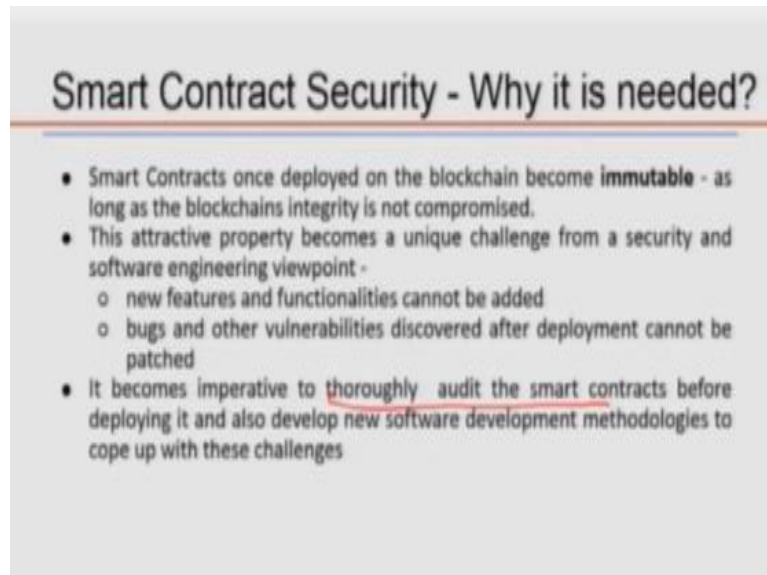
**(Refer Slide Time: 12:56)**



And that is what we have seen. So one of the major attack in 2016 that happened is called DAO attack. And we will talk about that a little bit in a bit. And since Ethereum is a public blockchain, all smart contracts are available. So you can study the security trends and patterns in the real world smart contracts, and analyze various smart contract verification tool on important contracts.

So we can do all this stuff. So it is not like I can just say, whether it is not secure, or I can say it is secure. I can actually take all the smart contracts that are actually deployed in the real Ethereum. And I can do analysis and I can show that there are problems. But I do not have to really show because there has been cases of cyber-attacks on the Ethereum blockchain. There has been attacks on the bitcoin blockchain and so on and so forth.

**(Refer Slide Time: 13:51)**



**Smart Contract Security - Why it is needed?**

- Smart Contracts once deployed on the blockchain become **immutable** - as long as the blockchain's integrity is not compromised.
- This attractive property becomes a unique challenge from a security and software engineering viewpoint -
  - new features and functionalities cannot be added
  - bugs and other vulnerabilities discovered after deployment cannot be patched
- It becomes imperative to thoroughly audit the smart contracts before deploying it and also develop new software development methodologies to cope up with these challenges

So in Ethereum, unlike bitcoin, the smart contracts are actually you know full-fledged programs, right. So therefore, and once they are deployed on this smart contract they are immutable, you cannot change them. And unless you do an attack on the integrity by compromising, let us say, 51% of the nodes and in consensus or whatever way you can rewrite the history, of course, then you can change the code.

But that is not something that is desirable, nor do you want that to happen. Therefore, once a smart contract is there, only way to stop that smart contract from being invoked in the future is to call what is called a selfdestruct function. Selfdestruct function reverts the money that is in the balance of the smart contract to the owner of that smart contract, and then it will self-destruct in the sense that nobody can, in the future invoke that smart contract.

The code will be still there, but it has already changed state to a self-destructed state. Earlier it was called a suicide function. Now the suicide function has been deprecated. So in Solidity there, you can create selfdestruct function. And that is the

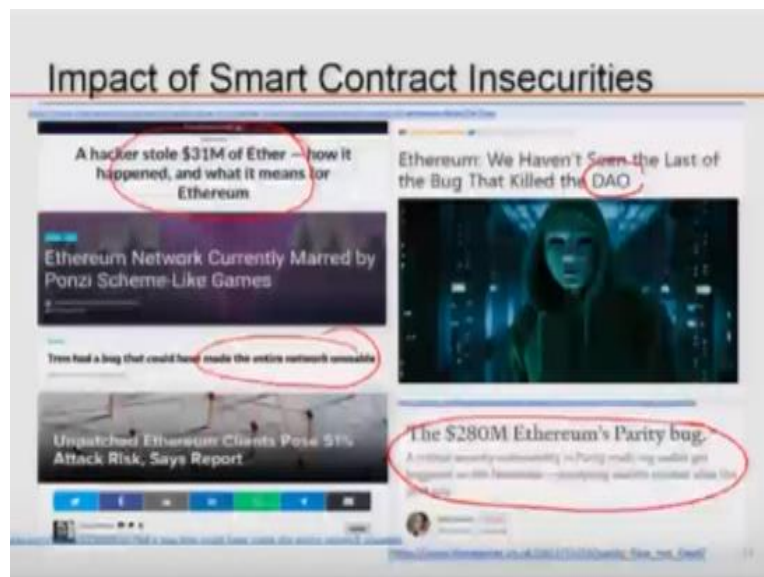


only way to stop a smart contract that you know is buggy to stop being invoked. But otherwise it is immutable.

So if you are otherwise happy with the smart contract, it is a pretty complex one and it has balance and so on. And you may not want to selfdestruct in that case, you know you have to live with the bug. So that is not good. So while being immutable is attractive, this becomes a challenge from the security point of view and software engineering point of view, because we are used to fixing programs and then do patching and re-install program when a new version comes out and so on.

But in this case, this is not possible. And also you cannot add features because then you have to rewrite the smart contract and then you have to redeploy it and then there will be two versions, and there will be a problem. So therefore, it is important to thoroughly audit the smart contracts before deploying it and develop new software development methodologies to cope with these challenges.

**(Refer Slide Time: 16:19)**



So why are we talking about security in smart contracts. So there has been number of different attacks on smart contracts. So one thing that happened was the DAO attack. So DAO is a distributed organization. The idea is that all the participants in DAO will put in ether in the smart contract. And then there would be a proposal by somebody who wants to be so it is kind of an investment bank type of smart contract.

So the initial participants will put in ether as investment. And then let us say I have a proposal to create an Ethereum application which will make money. So what I will do is I will submit a proposal. Then all the participants who invested in the DAO would actually vote on that proposal. And if the proposal gets a quorum of votes, I do not remember the number. But there is a number, I guess something like 20% or so then that proposal may be funded.

And the way to fund it is to create a DAO. It is called a child DAO, which will be given transferred, which will be a smart contract to which this sorry that amount, the proposed requirement, amount will be deposited and the signatory of that smart contract the owner of the smart contract will be assigned to that person whose proposal has gotten the votes.

So this way the original investors will be now investing into an enterprise and then if the child DAO has makes ether profits, then that will be proportionately distributed on the investors, right. So that is the that was the whole plan. Now the DAO had certain rules like for example, the smart contract had some lock in times.

For example, when you make a proposal, then you have to wait for a certain number of days in on during which there will be debates and among the investors which is off chain. And then there will be actually voting. And then the money will be spent, money will be sent. Now the money that gets sent to the child DAO cannot be encashed. It is also written in the smart contract until, I think 28 days or something.

So it gives you a time for this. Now what happened is that one of the participants, he found that way to create a child DAO and transfer money without having to go through a proposal and vote, and he did that. And then there was a big discussion and he claimed that this since the smart contract allowed it and philosophically really the whole idea of this smart contract is that it is a legal, the code is the law.

And therefore, if the code is allowing it, then it should be allowed. But that was not accepted by the original investors and there was lot of discussion. Ethereum Foundation got involved and then there was a question of what to do. So one, there were three proposals. One is not to do anything.

Second was that do a soft fork by putting in some new rule so that you do not have to you cannot do what you can do but then this may not be retroactively active unless you actually go and undo the past, undo the past which would be against the philosophy of blockchain because your past should be immutable. And third was a hard fork. And eventually a hard fork was done.

And so the in the new fork the amount of money that went to the child DAO by the attacker was reverted back. But, and the reason why this could be done is because of those lock in time. So the Ethereum community had a enough time to do all these arguments, debates, make choice between these three different choices.

And then eventually made this decision to do the hard fork because they was the 28 day lock in period in which the child DAO, the person who created this child DAO could actually had to wait for 28 days to encash. So this was somehow saved. But that created a hard fork in the Ethereum blockchain. So this was a protocol bug, in a sense that this was allowed somehow in the code, and not a bug in the sense that it was not a bug of integer overflow or underflow.

Or it was not a bug like reentrancy bug and stuff like that. So then there was a Parity bug, which basically, Parity is a company that created this particular smart contract. It is a multisig wallet. And in this wallet, what happened is that it used a library and this library basically had no owner associated with it.

So what the attacker did is that he figured out that in this library if I do an init owner and make myself the owner and then when a library function is called, what happens is that the smart contract which calls the library function, the library function executes in the context of that particular smart contract which called it. And therefore, the owner of the smart contract, that is that multisig wallet got changed to this person.

And therefore, he could drain out money and then what he did is that he then called selfdestruct on the library. So the library got self-destructed. So therefore, any multisig wallet that was dependent on this library was stuck. So they the people who

had ether balance in their multisig wallet could not get to their ether. And it was a big mess, right? So this was the Parity bug.

There are other cases where hackers stole \$31 million worth of ether. And then another case where there was some other thing that made the entire network unusable. These stories will take a long time. So what I have done is that when you get your slides, you can click on this links under the story or above the story. And you can go to the particular news item, which kind of describes all this bugs.

But the point is not that we have to know the exact details of this bugs. But what we need to know is that we have to make sure that the protocol is designed correctly and without bugs. We have to make sure that the code has no bugs. That could be actually exploited by an attacker, and so on and so forth. And that is the point that I am trying to make.

The second point I want to make, a very important point that this misunderstanding about blockchain that blockchain means secure is totally a myth. It is as secure as the person's code, the implementation who code the who create the protocol and who code the smart contracts and so on.

**(Refer Slide Time: 24:21)**

**Problems faced by Developers and Users**

- Since smart contracts is a new area of research, and industry is leading most of the efforts, not much information about smart contract vulnerabilities, tools and practices is available in the organized domain.
- The biggest problems are -
  - Studies on smart contract vulnerabilities are often unorganized and incomplete
  - Security tools do not have a corresponding academic paper
  - Most tools demonstrate their capabilities on different (sometimes random) datasets.
  - Also, they call similar vulnerabilities by different names
- This makes it impossible for a smart contract developer, or the end-user to make an informed decision about which tool to use and trust for their particular use-case.

So the smart contract security has been studied in bits and pieces by many people. And some of the problem that the so if you are a smart contract designer, implementer. So you have to know about these tools and techniques that will allow

you to validate that your contract has no obvious bugs. That does not mean that it will be 100% secure. But at least the obvious bugs should be ruled out.

So some of the studies in the past are kind of unorganized and incomplete. Security tools do not have corresponding academic papers. So the details of what it is actually capable of, what are its limitations, what algorithms it is using, what techniques it is using may not be clear. And most tools demonstrate their capabilities on different data sets.

So therefore, if I am going to go and decide that I am going to use this tool, I have to be able to compare against the other tools that are available, but there is no really benchmark on which all the tools have been tested and tried. Therefore, the developer will be kind of directionless on this. And also different researchers have called the same vulnerability by different names.

So that creates a, you know confusion as well. So we wanted to actually fix some of these issues with our work. And when we come back, we will talk about what we have done to bring some order in this situation. Thank you for your attention. And we will come back and we will continue on this.