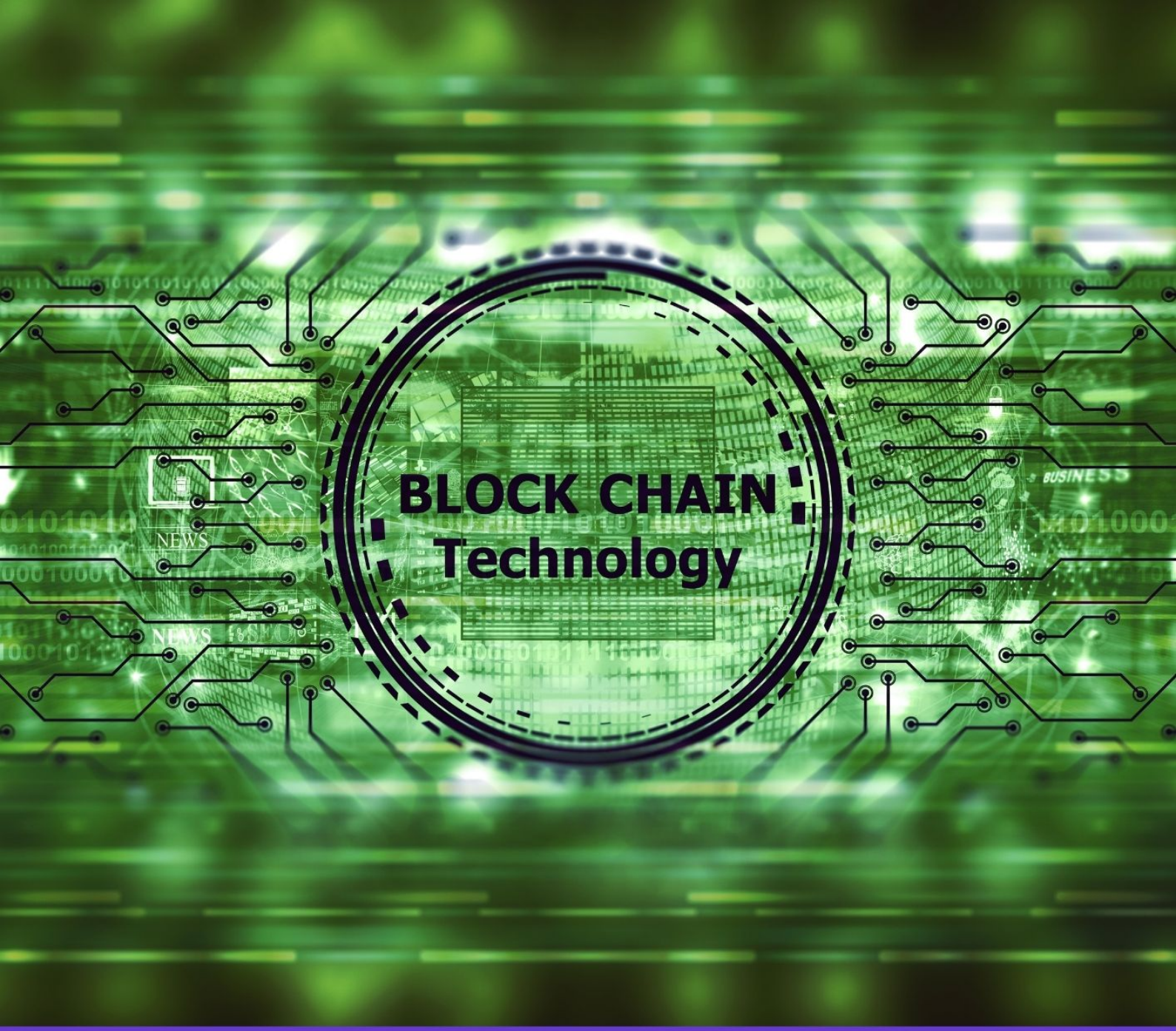


Introduction to Blockchain Technology and Applications

Prof. Sandeep Shukla
Computer Science and Engineering
IIT Kanpur

INDEX

S. No	Topic	Page No
	Week 1	
1	Lecture 01	1
2	Lecture 02	51
3	Lecture 03	108
4	Lecture 04	160
5	Lecture 05	173
6	Lecture 06	188
7	Lecture 07	205
8	Lecture 08	221
9	Lecture 09	233
10	Lecture 10	248
	Week 2	
11	Lecture 11	264
12	Lecture 12	285
13	Lecture 13	300
14	Lecture 14	315
	Week 3	
15	Lecture 15	331
16	Lecture 16	343
17	Lecture 17	361
18	Lecture 18	378
19	Lecture 19	391
	Week 4	
20	Lecture 20	405
21	Lecture 21	422
22	Lecture 22	432
23	Lecture 23	445
24	Lecture 24	467

Week 5

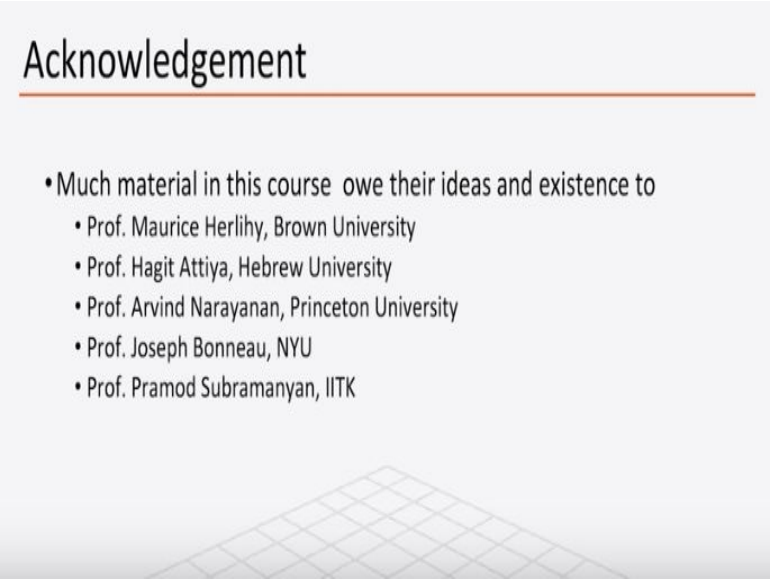
25	Lecture 25	483
26	Lecture 26	515
27	Lecture 27	539
28	Lecture 28	572
29	Lecture 29	595

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

Lecture - 01

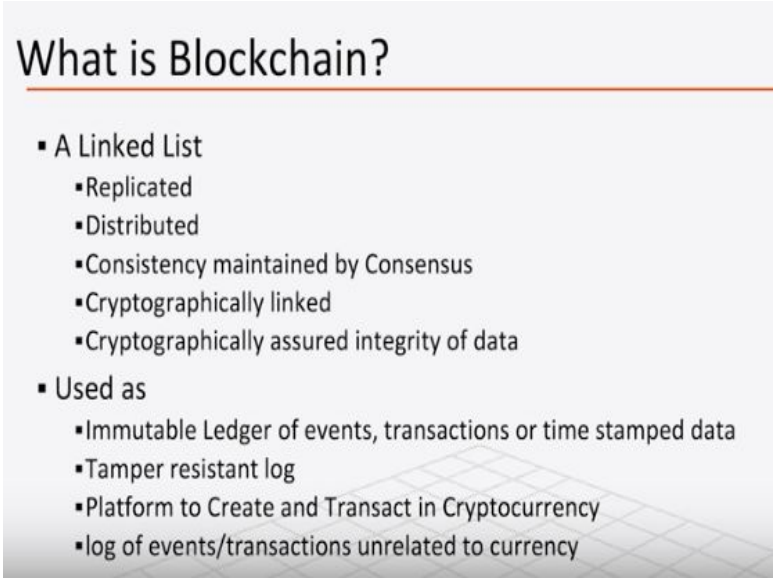
Welcome to the Blockchain Technology and Applications course. I am Professor Sandeep Shukla from IIT Kanpur and I will be teaching this course. So this course, the material on blockchain today is quite commonly available at various you know universities and various conference lectures and so on.

(Refer time: 00:37)



So I must acknowledge different colleagues from various universities whose lectures and whose material and books I have used to prepare the material for this course.

(Refer time: 00:51)



So what is a blockchain. If you think generally, a blockchain is a linked list. So what is a linked list? All of you know that a linked list is a set of blocks which are connected to each other by some kind of a link. In case of a data structure linked list, you have the nodes and nodes are connected by pointers and pointers are basically memory addresses.

But, in case of a blockchain the blockchain is a distributed data structure, which is replicated at various nodes or various computers and therefore, the linking is not based on memory addresses. So we have a different notion of linking between nodes and each of these nodes are called blocks. So therefore, you can imagine a blockchain as a series of blocks and each block is connected to its previous block by some kind of a link.

So and it is replicated all over because replication gives you number of different advantages like if one of the replica gets corrupted, the other replicas are there to make sure that the integrity of the information contained in the in the data structure is maintained. And also replication gives you some kind of guarantee of integrity of data.

And it is distributed in the sense that the the different computers involved in the blockchain platform actually are running distributed algorithms in order to maintain

the data's consistency and integrity. And the consistency of the data is maintained by a process called consensus. Consensus means that everybody agrees that the data that goes into the data structure is what they agree to put there.

And the linking as I said before, traditional link list the linking is through memory addresses. But in this case, we cannot use memory addresses for linking. So there is a cryptographic technique called hashing. So we actually use something called a hash linking and the integrity of the data is maintained because of cryptography techniques and consensus and replication.

Therefore, blockchain is a data structure that is maintained distributedly and that is replicated and main purpose of blockchain is to maintain the integrity of the data. And what is the integrity? Integrity means that the data once it has been agreed by all the relevant parties to put in the data structure, it has not been tampered with . Nobody has come and changed the data and and claim that this is the data that was put in.

That is made virtually impossible in a blockchain and that is the main property of the blockchain that it maintains the integrity of the data and as we will see that most of the applications where blockchain is used be it cryptocurrency or be it some other application, the integrity of the data is the main thing in blockchain. So what is blockchain used for? So first of all, you know many times we keep logs of events, right.

So when somebody accesses your computer, the computer keeps a log of the user names and and how they authenticated themselves. Microsoft Windows gives event logs every event that happens like you open a new program on your machine or something crashes or any any kind of event or you know it, you get connected to the internet. All these events are kept in event logs. So logs are very important.

When you do banking transaction bank keeps logs of when you interacted with its banking servers and what you did, what transactions you made, all this are logged. The main problem with keeping logs without any notion of protection of the integrity

is that somebody can tamper with the logs and somebody can delete some of the accesses. And therefore, later on when you check the log, you would know some part of its history.

So therefore, the blockchain is designed in such a way, so that it is an immutable ledger of events, which means it is a log that cannot be changed by a malicious party or by mistake. And therefore, all the data that you put in there could be event logs, it could be transactions, it could be various kinds of accesses and and modifications you do to some other thing like a data or you do a property transaction.

All these things logs are has to be kept in an immutable ledger. And that is what blockchain provides. So it is and the tampering of this data is made virtually impossible. So we do not say it is impossible to tamper, as we will see, as we learn more that there if you have a very high computational power, which is almost impossible for individuals together.

But if you can gather that kind of computation power you can actually subvert and this all the protection and change but since it is virtually impossible, we would say that this is a tamper resistant log. And therefore, having these properties, we basically use blockchain as a platform to create and transact cryptocurrency. So cryptocurrency as we will see bitcoin ethereum, these are cryptocurrencies.

And we will see that one of the first application of blockchain was bitcoin. So the whole idea of creating currency, that whose transactions whose creation whose use everything has to be put in a tamper proof log, and without a trusted third party or without a central agency, which keeps track of this logs and it will be clearer as we go into the course. And also you know many people confuse or conflate the idea of bitcoin and blockchain.

And as we will see through the course, that cryptocurrency is just a part of the story. And there are any number of other applications in which we need to keep tamper

proof or tamper resistant logs and there blockchain is a very good you know platform to use.

(Refer Slide Time: 07:50)

So why do we need a course on blockchain, right? So so it seems like you know if you look at the news and if you read mostly you know technology news, you will see a lot of news on on what is happening in bitcoin. Its price is going up and down, various kinds of other issues that come up. Sometimes there are cyber attacks on cryptocurrencies or sometimes people loose cryptocurrency for various reasons or cryptocurrencies are being used for illegal activities.

Ethereum is another such cryptocurrency. Now you will also hear a lot of news like for example, you will hear that United Emirates is fully going on with blockchain for their most of the E-governance applications. Governance means that you know all kinds of things like property registration, and you know car licensing, or you know license, driver licensing all kinds of things they are doing on blockchain.

And many other countries are doing the same thing. Same thing you will hear a lot about supply chain management on blockchain or you will hear energy management especially in case of micro grids, and renewable energy, you will hear a lot about the applications of blockchain or or you know electric vehicle charging stations and and

and paying for the charge, you know electric charging and all that on a blockchain like in Germany.

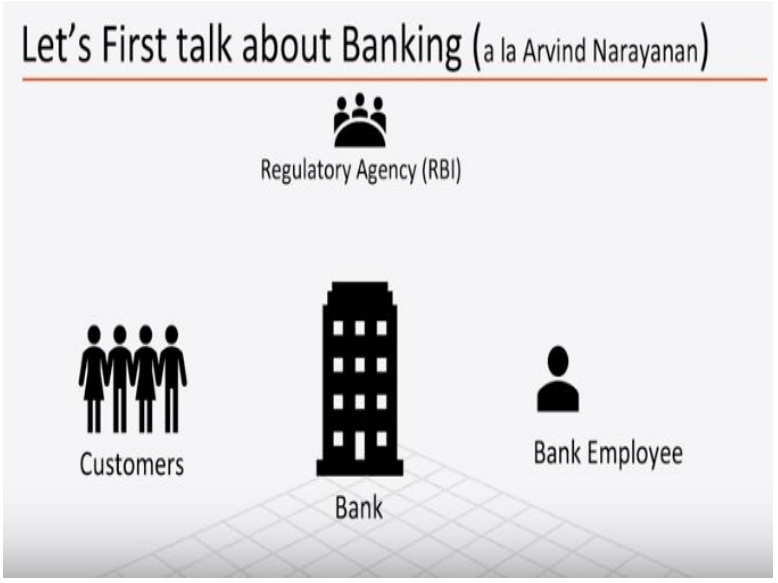
So very soon you will hear that the blockchain is for everything, for Nirvana. So people often say that okay, so this is a new hype and a lot of hyperbole. So what we will see in this course, is that it is not a hype, and it is a technology, quite transformative technology. So many people compare it with the advent of the Internet in transforming our lives and digitalization.

And similarly, people say that or predict that blockchain is going to be as transformative in the way we digitalize our functioning and our governance and our industrial dealings or financial markets and so on so forth. So even if you do not care about cryptocurrency and its market volatility, which is one of the biggest criticism of cryptocurrency, you will see a lot of application of blockchain which has nothing to do with cryptocurrency.

And this course, will try to teach you why that is the case. And in fact, I personally is a strong critic of cryptocurrencies like bitcoin and others. And therefore, my interest is that people learn that even if you do not care about cryptocurrency, you should care about blockchain as a technology, because it is a transformative technology. So this course is about that. So let us start with some examples where we can use blockchain.

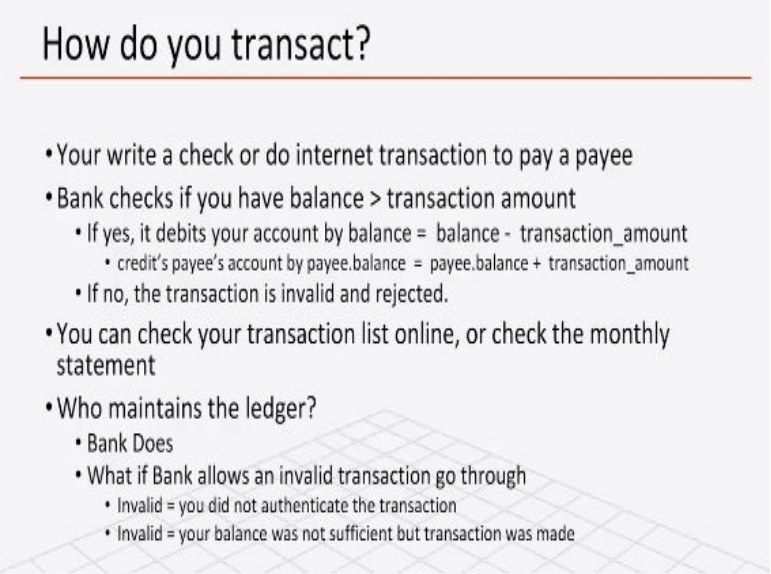
So I will first discuss some problems that you can probably solve with blockchain, and I am not going to initially talk about how we can solve it with blockchain because we have not taught you how the blockchain works, but I will tell you what the problems are with our traditional way of digitalization and and probably hint at you know maybe blockchain might provide a solution. So this example is by you know from Arvind Narayanan's book.

(Refer time: 11:26)



So so we will discuss how we do banking today, right. So in a bank banking system, we have multiple players, right. So we have customers ourselves, we have the bank and we have the bank employees and we have a regulatory agency. In case of India it is the RBI.

(refer time: 11:48)



So what happens is that when you do transactions, right. So there are multiple ways you do transactions. So let us consider two of them. One is you write a check to give money to somebody or you do an internet transaction like you login to your banking server and then you authorize the transaction and it goes through.

What the bank does and it does automatically not necessarily through intervention of human employees is that it checks if your balance is greater than your transaction amount. If it is not, then it will say it is an invalid transaction, it will reject the transaction. If it is yes, then it will transfer the transaction amount from your account to the other person's or payees account and your account will be debited by that transaction amount.

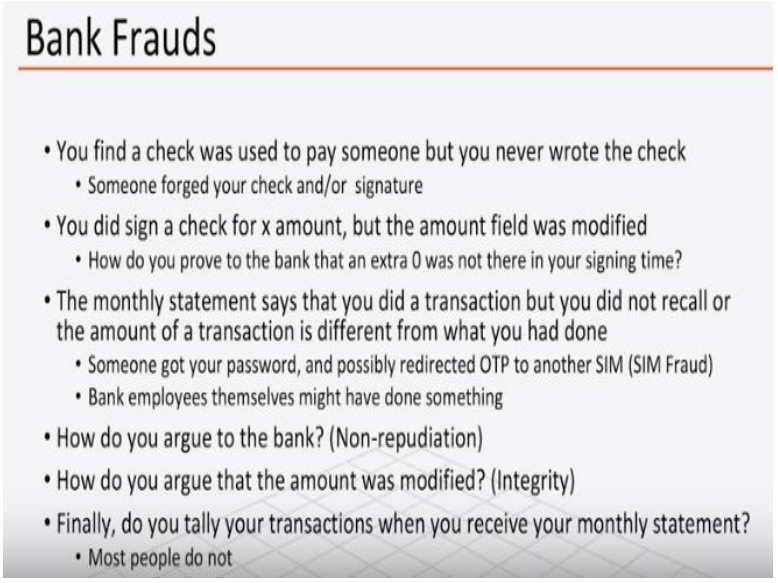
Now how do you know that this is done properly and correctly? You check your transaction list online or you get your monthly statement and then you check that all the transactions you made are recorded correctly. Now the question is who is maintaining this ledger from which this monthly statement is generated. The bank does. Now because of regulatory framework that we have, we do not see banks doing something wrong in this case.

We never, almost never find that, you know I made a transaction. And it was it went through. But now in my statement, I do not see it, or I do not see a normally do not see that I never did a transaction and it is appearing in my, in my statement. This kind of things did not, does not normally happen. But it could happen. How could it happen? If the ledger is wrong.

Or if somebody inside the bank tampered with the ledger and put a transaction in there, which was not made by you. In those cases, we say that it is an invalid transaction. And you say that I did not authenticate and authorize this transaction. Or your balance was not sufficient and the transaction still went through and you do not have a overdraft account. In such cases, we say that the transaction is invalid.

Then you go to the bank, and in most cases, the bank will be very surprised and they will not know, they have to investigate. And all kinds of stuff can happen.

(Refer Slide Time: 14:14)



In case of checks also somebody can forge your check. It recently happened to one of the top medical educational institutes recently. About 12 crores of money was transferred through forged checks. And those checks were checked under UV light by the banking authorities and then they turned out to be authentic checks but the institute said that it did not do the transaction.

Or what if an extra zero is put at the end of the amount field and somehow tampered with. So monthly statement will claim that you have done it, but you say that no. And another possibility is somebody obviously got your password and somehow fooled you into revealing your OTP or did the same fraud to get your OTP instead of you getting them. And in those cases also you would not know what is happening.

So it is a very difficult for the argument with the bank. So if you go and say that if you repudiate that I have not made this transaction, the bank may not believe it, because it will think that you are, you have no proof that you did not do it. So it is called non repudiation, you do not have the non repudiability, If you argue the amount was modified than the amount that you actually transacted, then you questioning the integrity of the data that is being now presented to you.

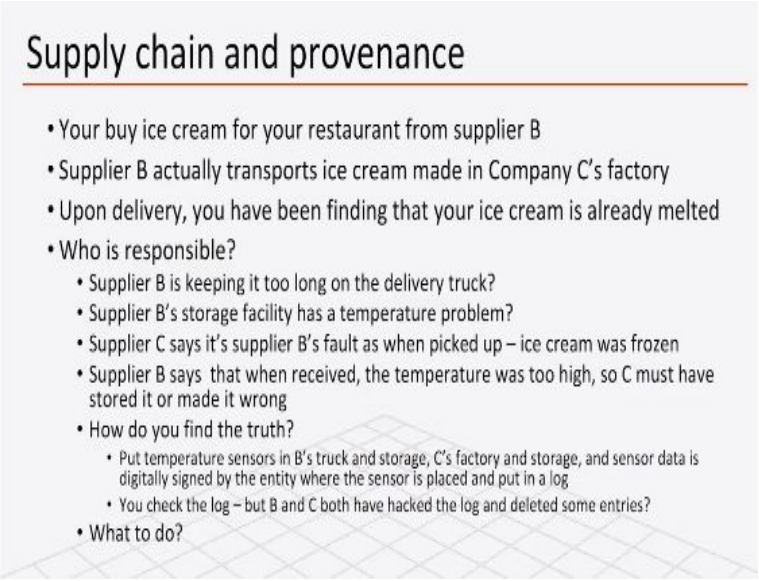
And there also you if you do not have a proof of what the original amount then your word against theirs. And finally, do you even tally the transactions like if you add the

entire debit and credit when you get your monthly statement. Most people do not do and and therefore, we depend or we trust the bank on all these issues. So the way the bank maintains its data about your transactions is not very transparent to us.

We trust them to do it and we trust them because they they are answerable to the regulatory authority RBI, and therefore we trust them. But that is what we call by a trusted third party enforced trust. And that is often a problem in in case of rogue banks and, and things like that. And so we are advocating here is more transparency in the way the transactions are done and transactions in records are maintained.

And as we will see, that the blockchain could be one of the solutions which will provide transparency and also maintain the integrity of the data and therefore provide more reason for us to trust the banking system without a trusted third party enforcing the trustworthiness.

(Refer Slide Time: 16:56)



Now second problem I want to talk about here is the supply chain provenance. So what happens in a supply chain is that you order something your business and you order something from another supplier and that supplier also orders various parts from a third supplier and the third supplier orders various parts from a fourth supplier. So this is what is called a supply chain.

And when the final product is made you as a business is making product whose quality depends on the quality of the components and other things that you have got through the supply chain. And therefore, if your quality has a problem, maybe it is not your manufacturing problem or your construction problem, but it is somewhere down the line in the supply chain and you have to attribute whose fault it is.

And to do that, you have to have proper transparency about where all the components are coming from. Normally, we only know the person who supplied us but that supplier also dependent on a third supplier, third supplier dependent on fourth supply. And we do not know any of that most of the time unless we go and investigate. And also there are quality guarantees that are not enforced from your end at the every step between two consecutive suppliers.

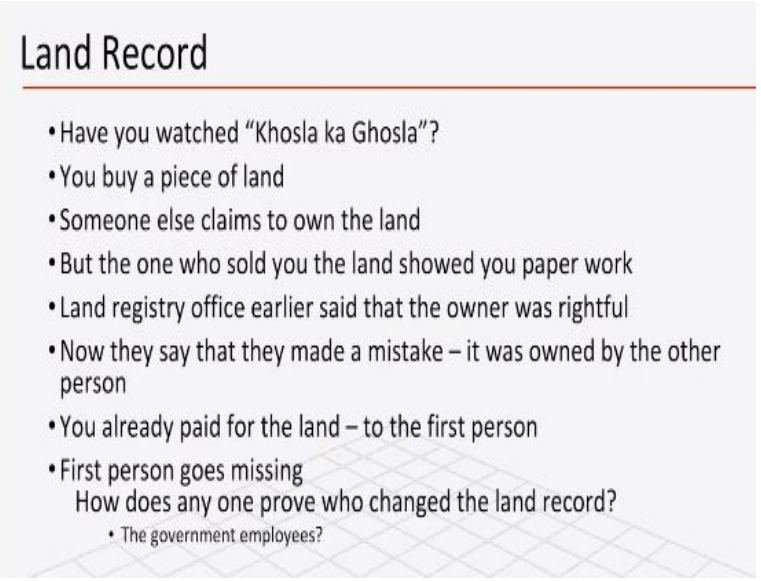
So I will give you a very simple example. This is an example I heard from Maurice Hurley. So suppose you are restaurant and you are buying ice cream from supplier B. Supplier B is not manufacturing or producing the ice cream but it is actually supplying from company C's factory. So to your customer, you are responsible for the quality of the ice cream, but the ice cream every time you are getting seems to be melted.

So who is responsible? So you can say to supply B that you probably kept the ice cream too long in your truck or your temperature has a problem in your truck. Or you can say supplier C, you made melted ice cream when it was picked up by supplier B, it was already melted. And supplier B says that when it received, the temperature was too high and therefore, this is the problem. So how do you find the truth?

So one possibility is that as a restaurant, you can put temperature sensors on the truck of supplier B as well as in the factory of supplier C where it stores its freshly produced ice cream. And then you get the sensor data from the suppliers directly through wireless network. And then you keep a log and you check the log, you would know whether B or C had a bad temperature and that is why it happened.

But there are certain problems and we will talk about that. We will come by, revisit this problem about the integrity of the data that is coming from the supplier B and supplier C even when you actually put those sensors, and we will see how maybe the blockchain might be a solution. A third problem here is Land Records.

(Refer Slide Time: 20:12)



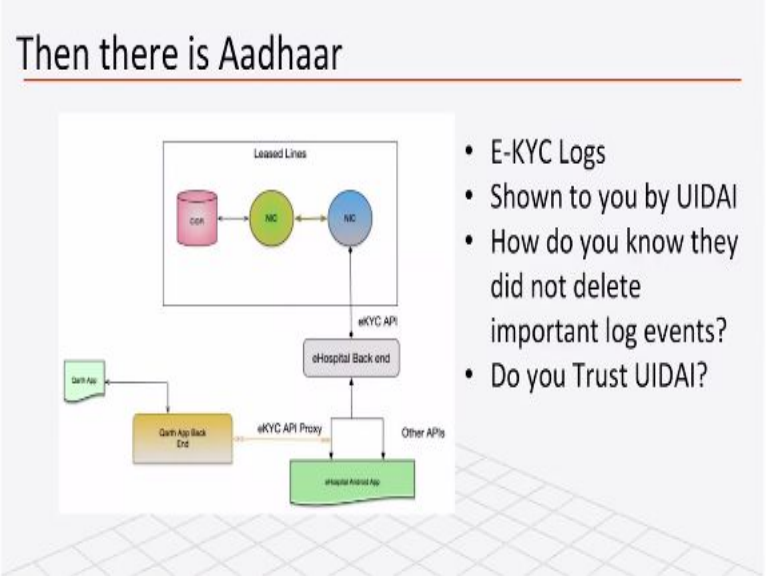
So many of you have watched this movie, "Khosla ka Ghosla" right. So so you buy a piece of land and somebody claimed that he owns the land and showed you some paperwork, and you paid him or her the money and land registry office, you checked and land registry office said yes, this land belongs to this owner. So you trusted that, and you went ahead and paid. And then this person disappears.

Then a third person comes and says that this land actually is mine. And so sale is not legitimate. So at this point, you go back to the registry and registry says oh, we made a mistake. Actually, some data was missing. And now we know that this person is right. Now you are you basically have no land even though you paid somebody who is now missing. So somebody might have changed the land record twice.

Once in the name of that other guy and then name of this guy and all the logs for this change of this data has also gone missing, because we are trusting the land registry authority to maintain the integrity of the land record. And also if any access is made for changing any of the land records, maybe the logs has to be protected also but the

logs were not protected properly, because it is an insider who has managed to delete relevant entries in the log so it is not clear who accessed the log.

(Refer Slide Time: 21:42)



Also there has been a question about your Aadhaar accesses. So remember that there was a case where people who did the using Aadhaar authentication, they did their KYC, or getting SIM card from Airtel. The Airtel bank also made an same E-KYC to create an account in your name and then all the you know cooking gas subsidy was being now deposited to that bank account.

Now this is actually this was shut down later on this process but we need to know who is accessing our data to the in the other database for doing E-KYC. So then UIDAI decided that through their website, you can go there and see who accessed your Aadhaar information or who did E-KYC on your this thing. But it did not show the demographic lookup properly but it showed the E-KYC logs. But this logs are now maintained by UIDAI.

And if you if you trust UIDAI, then you will say okay, these are the only times when somebody did E-KYC on me. But if you if you think that somebody is doing E-KYC and somebody inside the UIDAI might have deleted the logs, there is no way to prove or disprove that. So there is a trust problem that can build up.

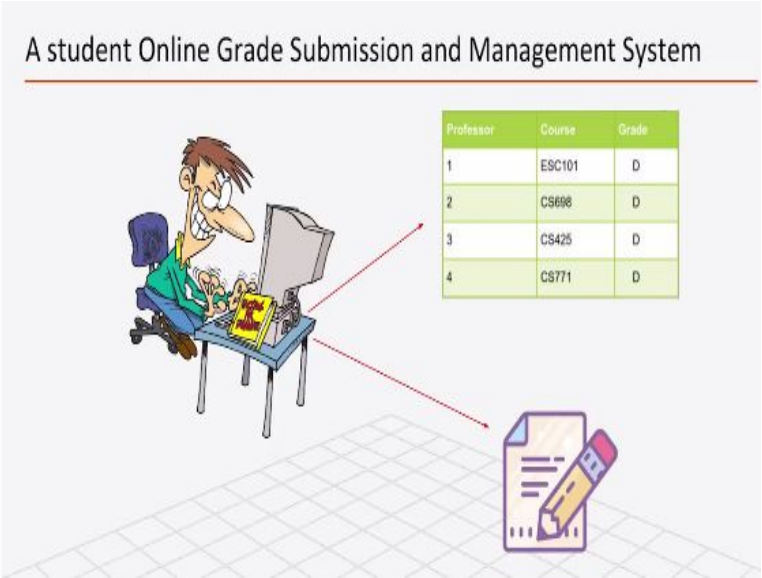
So therefore, you know in this case also the tamper proof property of this logs, should be somehow convincing to the customers or to the users so that they can trust that whatever they are getting as log of their E-KYC is indeed the only times when their E-KYC was done. So so now we have spoken about four problems. One is your banking transaction logs and their tampering possibility.

Second we talked about supply chain data and supply chain provenance and their tamper proof property. Third thing we talked about land registry and need for the integrity of that data and maintaining keeping the integrity of the logs of accessing and modifying that data. Fourth thing we are talking about the E-KYC logs maintained by UIDAI.

And in all these cases in the current traditional system, we have to trust a third party or we have to trust the the authority who is maintaining the data to actually maintain the integrity of the data. And that may or may not be very prudent to trust. So then finally, the great submission and management system in a university or in an institute, where professors submit grades online, and it goes to a database.

And that database is now you know maintained and then the students when they log in to see their grades or you know print out their grade card, they actually access the database.

(Refer Slide Time: 24:57)



Now between the time professor submits the data and when the student sees the data, between that time if somebody insider actually changes the grade, then there is no way for the professor to know because professor might have 400 students in the class and unless there is an investigation when he might have written records of the thing of the data.

But if nobody has complained and there is no way for the student to know unless the grade change has been very drastic, like somebody is supposed to get an A and got an F. But otherwise, there will be there will be no way to know that the data integrity has been violated. And the person who did this inside might also remove the log of accessing that database and that could be a problem.

And therefore, there is not enough transparency in which the data is maintained without you know leaking out the private information like the grades of students, but there should be transparency of how the data is maintained and and some kind of a proof that the integrity of the data has not been tampered with. And then only the there will be full trust from the students or from the professors that this system is tamper proof.

And there is no way for somebody to come and change the data from from inside or by through some cyber-attack or something like that. So these are the things where

blockchain might come handy as a technology to solve this questions that I have raised so far. So people might expect that oh, so this is a course on blockchain. So it might be about bitcoin and ethereum and we would learn how to actually transact and all that stuff, but that is not the case.

And here is why I would argue that why I do not want a course on bitcoin or currency. However, if you are in the full dynamics of bitcoin and and how it works and how you actually make transactions and how you can make you know micro payments and all that with bitcoin, then you can actually go to Coursera. And there is a very nice course by Professor Arvind Narayanan who actually from Princeton, and whose course is about bitcoin only, right? But our course is not on bitcoin.

(Refer Slide Time: 27:27)

And the reason is that bitcoin is something that has also raised a lot of concern about cryptocurrency. So one of the very early story after the bitcoin came into existence was the history of the Silk Road, Silk Road was in underground market and the underground in the sense that it was being carried out in the dark web. So dark web is basically is the part of the web where people do not have direct access.

Like we go to a website of a news organization or an institute or, or a business. These are places which are accessible through the web, but they are not accessible directly by common people. You have to know exact address and their location and probably

very strong authentication. And many times you cannot even go to them without using some anonymous anonymous browsers like Tor, Tor browsers.

And there the people from the underground operations of drug and other things extortions and all that stuff they actually transact in bitcoin and Silk Road was dismantled by FBI around 2013 or so. But they are still a lot of business in the dark web that happens using bitcoin because bitcoin is almost anonymous. There is no way if you ask me to send money to a particular bitcoin address, there is no way to know whose address it is even from the law enforcement agencies.

And therefore, it is very difficult to trace. And this is the same thing that is happening today when there is a ransomware attack. And ransomware attackers encrypt your entire disk by sending a malware through your phishing link or something. And then they actually demand money to unencrypt or decrypt your device by and they want you to pay in bitcoin or ethereum because of the anonymity that they can hide behind. So that is one of the problem of bitcoin.

(Refer Slide Time: 29:42)

So illegal activities and all ransomware these are some of the problems. But there are other problems, right?

(Refer Slide Time: 29:50)

So one of the thing you would know that bitcoin prices go up and down, sometimes they went to close to $20,000 a bitcoin and now I think it is about 12,000. Sometimes it was 5000, sometimes it came down to 8000. So if you think that investing in bitcoin is a good idea, that is not something that I would advocate. If you want to invest and quickly make money in in a couple of days, and you have a lot of money, maybe you can invest in bitcoin.

But otherwise it as a long term asset, I do not think the bitcoin is a bitcoin is a little too risky unless you have that kind of risk appetite you probably do not want that. The other thing is that the way bitcoin works. So we talked about the consensus mechanism, right? So we have not we will discuss this later in the class how the consensus works. There are many ways to arrive at consensus.

So consensus is when all the parties agree that this transaction indeed took place. And then only it goes to the ledger, right? So this consensus process in bitcoin because of its anonymity and the fact that it is anybody can join the bitcoin network. So therefore, we cannot tell whether people are honest. So therefore, the consensus mechanism has to be very, very cryptographically challenging.

And to solve those cryptographic puzzles, and we will talk talk about that later in the course. It requires the immense amount of computational power. And when you do

that kind of computation for arriving at consensus, and you actually have to arrive at many consensus over a short period of time.

(Refer Slide Time: 31:35)

And therefore the amount of energy that you require to do that kind of computation is enormous and that is very harmful for the environment.

(Refer Slide Time: 31:56)

And we will see that if you look at the statistics that bitcoin mining today, actually process of arriving at a consensus actually is also called a bitcoin mining The reason why it is called mining is because anytime a consensus is arrived, there will be only one winner whose selected data will be actually made permanent into the data store or in the blockchain and they get some reward.

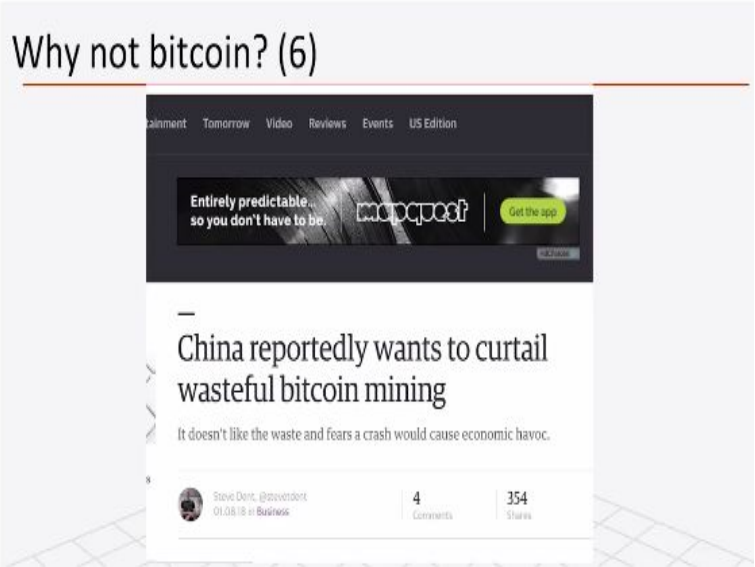
So in the beginning, bitcoin was giving 50 bitcoins as reward. If your data collection becomes permanent, then they halved it to 25 bitcoins and at this moment it is 12.5 bitcoins that you can get, but this is how bitcoins are also created. Therefore, this consensus process in bitcoin is also called bitcoin mining.

And this mining process is so so expensive in terms of computation and therefore in terms of power consumption, that it is consuming more electricity than 159 countries, including Ireland, and most countries in Africa. So you can think how wasteful it is to actually mine bitcoins. So and any other cryptocurrency that is also based on participation of public anonymously would have to have a similar kind of consensus process.

And therefore, they will also be wasting a lot of power and that is a big concern for from the environment point of view. Also it becomes very expensive to buy equipments, which are capable of doing such computation. And therefore, it get more tilted towards people who have deep pockets. You and I with our regular laptops or or other machines we cannot, you know have the ability to win this process of consensus.

So we cannot make bitcoins whereas the there are people who have invested millions of dollars in buying equipments and paying electricity bill and air conditioning bill for their huge compute farms, and therefore they are able to do this. So this makes this entire bitcoin ecosystem very very tilted towards those people.

(Refer Slide Time: 34:10)



And the other thing is that most of this bitcoin mining companies, now actually there are companies which actually can only afford such large-scale bitcoin mining and this many of these companies are in China. Now what happens is that it is it is provable, that if one person or one player in the bitcoin platform has more than 51% of the computational power, then he can actually completely change the data in the bitcoin network.

So earlier we said that in a blockchain, the data that we store after a consensus is tamper proof, right. Nobody can come and change the data. The reason why they cannot do that because to the amount of computation they have to do, in order to tamper it is too much because they have to then undo all the computation that was done to store that data and any other computation that was done to store data subsequent to that.

So this is an total amount of work, computational work that somebody has to do to tamper with. But if I have majority of the computing power in the entire ecosystem, then I can actually completely change the entire chain, chain of data, all the data I want to change and all subsequent data. Now think about this.

If one country has most of these or majority of this companies who are doing bitcoin mining, if they want to get together and then their total computational power becomes

above 51%, then bitcoin will have no integrity of its transactions. And therefore the entire ecosystem will fail.

Also because of this environmental concerns and volatility of the cryptocurrency and so on, we may be the country in which all these companies are there and they are consuming power, they are polluting the environment, the country might decide that they will curtail the activities of this companies and therefore, there will be another kind of problem. And when this kind of news has come, you will see the bitcoin prices fall.

The reason is, even though it may be a speculative news, because because we are now dependent on all these different extraneous things which are not imagined by the originator of the bitcoin and originator of the bitcoin was an anonymous person who used the name, Satoshi Nakamoto. And in 2009, he wrote the first you know version of the bitcoin and he launched it and he also wrote a white paper.

And his idea was to get rid of the tyranny of the large banks, right. So in 2008, there was a international financial meltdown and it turned out that the banks did a lot of things which are not properly you know correct things to do. And in fact, from the writing and from the original message that was in the first block of the bitcoin blockchain, it seems like he was actually trying to emancipate people from the, from the hold of the banks.

And then they want the entire world everybody to become part of a currency system, which is not dependent on a central bank or on the banks to actually decide the, you know the value of the money. But unfortunately, the way it worked out and way it unfolded is that only a handful of companies are now doing most of the bitcoin mining.

And then the all the transactions that are happening are also by only a you know only 1, 2% of the people who are involved in the bitcoin ecosystem. And the same thing is

happening in ethereum that 90% of the transactions are done by maybe hundred players when there are about 6 million players in the ecosystem.

So you can see that there is a huge amount of inequality that is building up in the cryptocurrency system when there are only handful of players who are doing all the activities, all the transactions storing all the all the currencies, and the rest of the other people are doing very little. And therefore, the entire dream of Satoshi Nakamoto whoever he is, is not being translated to reality. So that is another problem.

And as you know that in India, the RBI has put a blanket ban on bitcoin as a transaction medium and or any kind of cryptocurrency and there is a good reason for that, and we will not get into that, but that is how it is. So then there is another problem. That this problem has been happening because of various reasons.

(Refer Slide Time: 39:18)

So that so common people so bitcoin if you want to directly play into the bitcoin ecosystem, do mining, do transactions you have, you have to be computationally savvy, right you have to know how to create a private key, how to create a public key, how to tell other people that this is my public key which which basically get shortened to your wallet address.

And then you have to store your private key very, very securely in order to so that nobody knows your private key. Because once you know once they know your private key, the money is no longer yours because they can use your private key to spend the money or transfer the money to another account. So so this cryptography, the public key, private key is very important for operating in the, in the bitcoin or ethereum economy.

Now common people, businessmen, they might not be computationally savvy so they have another way of investing in the bitcoin or ethereum or any kind of cryptocurrency is by going to an exchange cryptocurrency exchange and say that I am giving this kind of money and give me this worth of cryptocurrency. Now what this company will do is give you some kind of a paper or some kind of a link that says that you now own so many bitcoins because you give me this amount of real money.

And then all the private key and everything will be stored at the exchange. So there there is a there are two kinds of wallets like hot wallet and cold wallet, and most of the keys that are not very frequently used are put into the cold wallet and and and and the hot wallets are the ones that are being used very commonly.

Now this happened in one of the exchange Mt. Gox where there suddenly the exchange owners, they said that there is a so many millions of worth of cryptocurrency has been transferred to somebody else's account and as I said that once it is transferred to an account, the account number is actually a public key in a in a hash form, and therefore no no way to know who is behind it.

So the police or anybody cannot really find the person whose wallet has got that money. So so therefore, the owner said that, you know the private keys were stolen, and therefore this happened. People suspected that the stealing person is one of the exchange owners only. And since the actual people who purchased bitcoins from this exchange did not keep their own keys, the keys were in the in the cold wallet of this or cold vault of this exchange.

So therefore, they have no recourse, then they could not get the money. And this was a huge loss for a lot of people. More recently, there was another case.

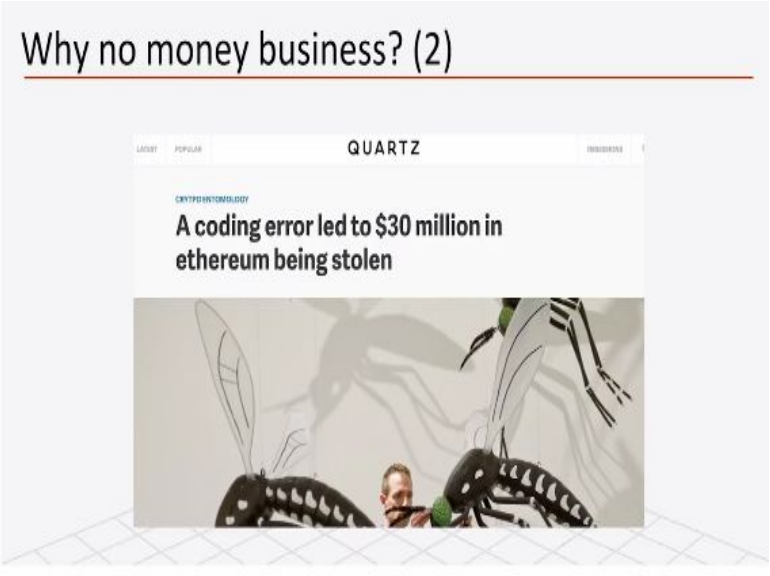
(Refer Slide Time: 42:24)

So this is the Quadriga is another such exchange. And in that exchange, what happened is that the person who owned the exchange also had the in his cold wallet, a lot of keys $145 million worth of cryptocurrencies keys he had and then he died. And with him the password or code for accessing that wallet, you know went away and nobody there was no backup.

Nobody else knew that code, and therefore $145 million worth of cryptocurrency was lost and the losers were people who actually bought them. And now recently the news is that they are exhuming his body to figure out if he actually made a disappearing act by killing somebody else like so they want to check whether this person who is buried as him is really him. So this is this has come to that.

So so this is the other issue about cryptocurrency is that if you are not tech savvy and you are not doing it yourself, you are dependent on this exchanges and all this happenings can happen to you and you may lose a lot of money.

(Refer Slide Time: 43:36)



The other thing that is also happens is that there are bugs in the code of the blockchain ecosystem, or in case of ethereum the bug was in the smart contracts. So smart contracts are programs that work on your behalf, from your address, from your account and do transactions and other things. And so in this case, there was a bug reentrancy bug.

And therefore, attackers could actually make transfer from all of this smart contracts from various people's accounts $30 million worth of ether to their accounts. Fortunately, some of the good hackers, they realize this is happening. So they actually also use the same bug to actually siphon off rest of the money of the rest of the accounts to their address and later on they paid it back to the rightful owners.

But the $30 million worth of ethereum, ether that was lost was lost to the hackers. So these are the kind of things that are concerning about cryptocurrency.

(Refer Slide Time: 44:47)

So in summary, I would say that, let us admit that bitcoin first brought to us the technology of blockchains. So it was a very good thing that happened. The Satoshi Nakamoto actually he understood how to create an ecosystem in which there is replication, distributivity, consensus mechanism, all this technology and algorithms that already existed in the distributed system literature.

He actually brought them together and also brought together the some some already existing concepts of digital cash and the idea of what we call no double spending and all that thing together, he brought them together and created bitcoin. So we are kind of like happy to have the technology with us, but bitcoin as such or or if you consider ethereum and other cryptocurrencies, we have seen in the last 10 years or so there is too much interest by investors to park their assets there.

And if that happens, then the whole idea of a currency, what is a currency, currency is something through which you exchange value, right? Now if you just park it somewhere then it is not being used as a currency. It is been being used as a storage medium of value. And so that is one problem. So it is not being used so much as medium of value exchange.

The private key stealing and private keys at exchanges is putting people to enormous risk of losing large amounts of money. There are vulnerabilities in the coding of the

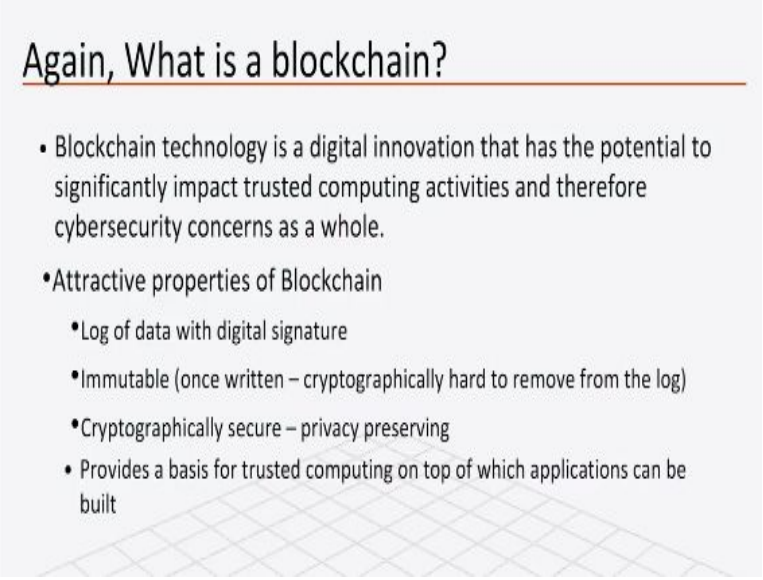
smart contracts and other parts of the code, which poses risk of losing money. There is also market volatility based on various news as I was mentioning, and therefore there are lots of possibilities of losing money. There is a lot of energy wastage.

We talked about that climate impact and too much concentration in one country, which is also a problem. And then there is regulatory risk because regulators of different countries are framing their rules for allowing or disallowing cryptocurrency or regulating them. And also criminal activities use bitcoin. So all this together basically makes us not to focus on bitcoin or cryptocurrency in this course.

Even even then we will talk quite a bit about the bitcoin and ethereum because of learning from them the technological underpinning, which is very useful and very worthy to understand. So we will talk about bitcoin, how it works, how the consensus works there. And and also we will talk about ethereum and smart contracts and all that. But we will not focus on this application of blockchain and cryptocurrency.

We will focus on application of blockchain for non cryptocurrency usage in this course.

(Refer Slide Time: 47:49)



So then, coming back to blockchain, now let us say blockchain technology is a digital innovation that has the potential to significantly impact trusted computing activities

and therefore cyber security concerns as a whole. So trusted computing is a is a concept that when you do computing and if you depend on the results of the computing and by computing we not only mean number crunching, we also mean the entire ecosystem.

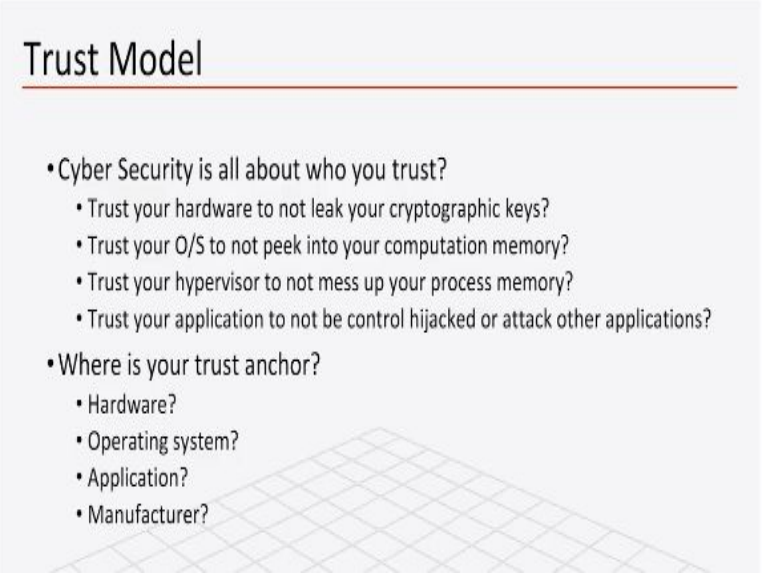
The the exchange of information over the network, the storage, the databases, all these things are we are clubbing together as computing. But when you do computing and your system depends like E-governance system or or your identity system or your your grade management system or your property registration system, that dependent on a whole computing ecosystem.

So but then if you are depending on the computing ecosystem, there are many different possibilities by which data in that ecosystem can be impacted and and tampered with and be it while it is transiting through the network, be it while it is in storage, be it while it is being computed. So therefore, we have to worry about trusted computing when you can trust the computing environment to produce result.

Keep the integrity of the result and give me transparency of how it is processing the data, how it is storing the data and how it is protecting the data against various kinds of attacks and that is what blockchain can give you. And the attractive properties of blockchain for this purpose is that it gives you a tamper proof log of data with authentication which means that the data also is signed by whoever created that data and then it it is immutable or almost immutable.

So which means that once written, it is cryptographically very computationally hard to remove that log or to tamper with the log. And if you want the transparency but not reveal the private information, you can do that in blockchain. So it is it can you can make it privacy preserving. And it provides a basis for trusted computing on top of which applications can be built.

(Refer Slide Time: 50:22)



So let us talk a little bit about when we said trust, what do you mean by trust? So trust model that you assume in case of computing environment, we have to consider what are the trust is kind of like opposite of your attack model. Like who can attack or how threat model like what are the threats to the integrity and privacy and availability of the information? So first of all, you can put your trust on the hardware and say that hardware will not leak your cryptographic key.

But is that true and it is not necessarily true because there is something called site channel attacks. So which people have shown that they can actually get your cryptographic key while you are doing encryption or you are doing decryption, they can look at the power that you are drawing from the your power you know source or they can look at the electromagnetic signature or they can look at the timing and all kinds of things by which they have shown that they can reveal your cryptographic key.

So unless your hardware is in a proof to that kind of manipulation, your hardware may not be trustworthy. So nowadays there is a lot of work on how to make a hardware trustworthy by using enclaves and other kinds of mechanisms and even there we have people have shown their ways to subvert the protections that the hardware companies come up with. You must have heard of Intel's meltdown and Spectre bugs that came out two years ago.

So so you may or may not be able to trust the hardware. So in that case, if you do not trust your hardware, you have to make sure that you do the protection against any kind of key leaking and all that stuff at the next layer which is the operating system layer or firmware layer. And there the question is, now operating system is basically create processes and they have memory in which the computational results and stuff are temporarily kept.

And this memory is also hierarchical. So which means that some of part of the memory addresses sometimes come to the caches, and and all that stuff happens. So therefore, the question is, are there ways in which people can peek into your cache or peek into your memory and then look, get your private information, your private key and all that stuff? So that has that is another question of trust.

Now if you you are using virtual machines, then you have to know whether your hypervisor leaks information, they can peek into your process memory and all that stuff. And then your applications may be also have bugs and there may be cyber-attack. And while the application is using some of your data, it may actually tamper with that or it may actually reveal private keys and all that stuff if your application is using your private key without any protection.

So the question is, what is your trust anchor? Right is it in the hardware? Or is it in the operating system? Is it in the application or you trust a manufacturer like Intel or others to actually give you a trusted platform?

(Refer Slide Time: 53:43)

Trust Model (2)

- In many real life transactional activities – trust model is the inverse of the threat model
  - Do you trust your bank to not take out small amounts from your balance all the time? (Watch – "Office Space")
  - Do you trust the department of land records to keep your record's integrity?
  - Do you trust UIDAI officials to keep your aadhaar data from unauthorized access?
  - Do you trust your local system admins to not go around your back and change settings, leak passwords, change database entries, and remove their action from system logs?
  - In the patch management system of your enterprise, are the patches being put -- all have digital certificates? Who put them? Do you trust your employees to do the correct thing and not put a malware as patch?

The but in case of the applications like E-governance and you know grade management system or whatever, the question is, do you trust the bank that they are not taking away small amounts from your balance all the time? There is a movie called Office Space that was taking only a you know fraction of a cent from every transaction, and very soon they accumulated millions?

Do you trust the land record department to keep the integrity of the land records? Do you trust the UIDAI officials to keep your other data access logs you know integrity of that? Do you trust your local system administrators to not to go around your back and change your data or read your data and then delete all that information from the system log.

If you have a patch management system in your enterprise, how do you know that the patches are being made, or somebody is doing an unauthorized patch to attack your system and things like that. So that those are the problems that you have to worry about. So so far, we have spoken about the trust model and corresponding threat model which are kind of inverse of each other.

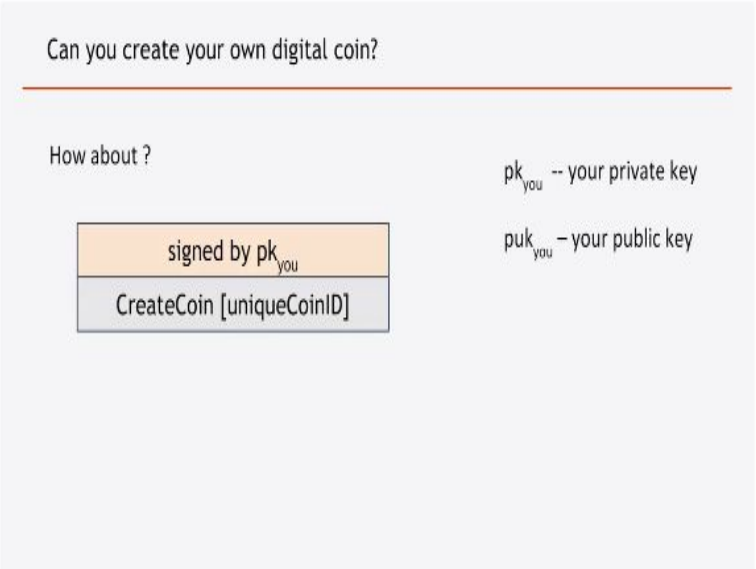
(Refer Slide Time: 54:55)

Now let us look at some of the cases that we talked about before in detail in light of what we have discussed since then.

(Refer Slide Time: 55:05)

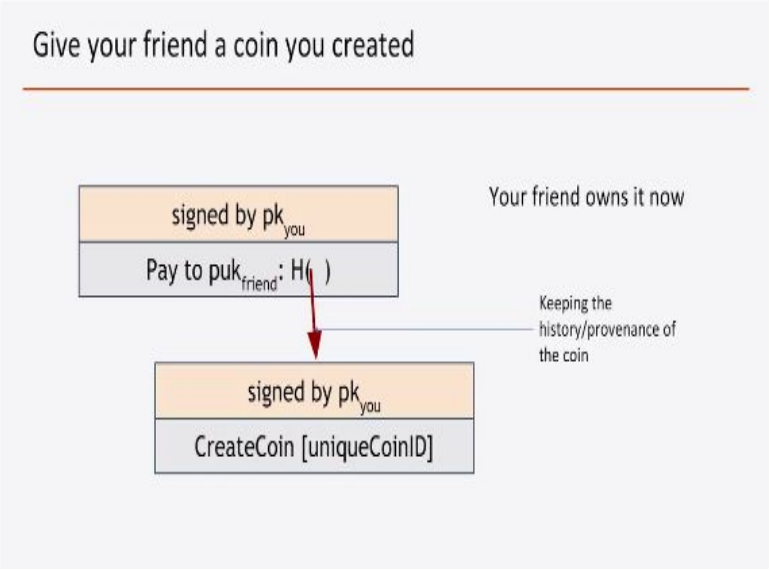
So let us first look at how one can create a digital coin. And this is Arvind Narayanan's crypto coin version 1.

(Refer Slide Time: 55:15)



So what you can do, you can say that I am going to be your mint, and I am going to create a digital coin. And then I will give it a coin ID. And then I will have a public key and private key and I will sign the coin with my private key. Okay, so anybody who has my public key because public keys are usually known to everybody else, so he can check that it is my signature and once it has my signature it is a valid coin.

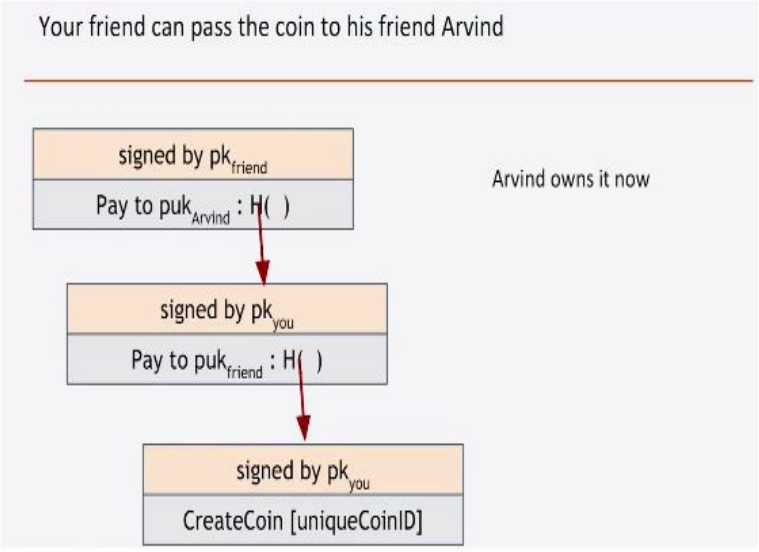
(Refer Slide Time: 55:50)



So then, so I can then give the coin to somebody else. Maybe in exchange of some value, you know in the real world. Like I buy a book from him. And then I give it to a friend. So how do I give it to a friend? The friend creates an public and private key. And I give the friend I say that pay to his public key address this coin, and I sign it with my private key. So anybody can check whether I signed it by using my public

key. And I also in order to keep the history of how this coin was created, and who gave it to whom I created a link to the previous information that the coin was created by me. So this red link that you are seeing here is actually that link that is keeping that history along.

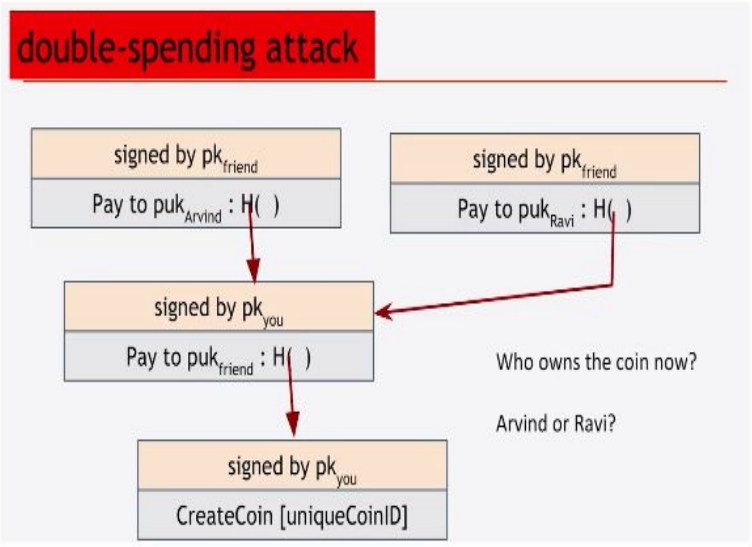
(Refer Slide Time: 56:43)



Now since I have now the coin I earned it by giving you a book or something selling you a book. So now I want to give it to Arvind who now gives me something like maybe another book. So I want to give the money to Arvind. So I say that pay to Arvind's public address, public key and then I sign it, right. And I am the friend. So I sign it. And then I keep a link to the previous to previous link to information, pieces of information.

So if anybody wants to check where this coin came from, then he has to basically go down the links and see where the coin was created, who got it first and who got it second, and so on. So this is way so every time I pass the coin, all this information is also passed about the coins history. So Arvind now owns the coin.

(Refer Slide Time: 57:35)



But if I am dishonest, what I can do, since this is digital, so I have no problem to pass it on to multiple people. So I actually give a copy to Ravi, another friend for another thing that I get from him and do the same linking and all that. Now the question is I had only one coin. Now I gave it to two people.

Now these two people will have to you know both have the same coin that is not possible right because if you have a real world coin, then you could not have given to two people since you are digital you are cutting and pasting. So you can actually give it to two people. And this is what is called a double spending. So I have done a double spending, which cannot be done in a real money monetary system.

Because then, you know if I can do double spending, I can do triple spending, and then the value I got from this one coin is twice the real value, right? Because I got something from Arvind I got something from Ravi, but I am just giving them the same thing. And they also when they try to use it, there will it will be a problem. So this is called a double spending or since it is actually an attack, call it a double spending attack.

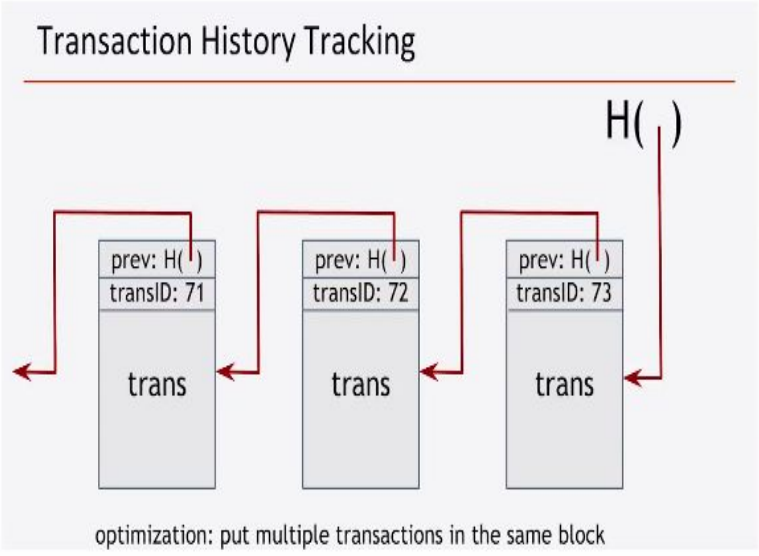
(Refer Slide Time: 58:56)

So major challenge has always been in the digital currency. From 80s people have been trying to solve this problem of creating digital money and this all this people have come to this problem that how do you stop double spending. And if you can do double spending, you can do triple, quadruple all kind of spending.

(Refer Slide Time: 59:18)

So we have to do something more than what we just showed you in the version 1. So version 2 is how we can make a double spending proof digital currency.

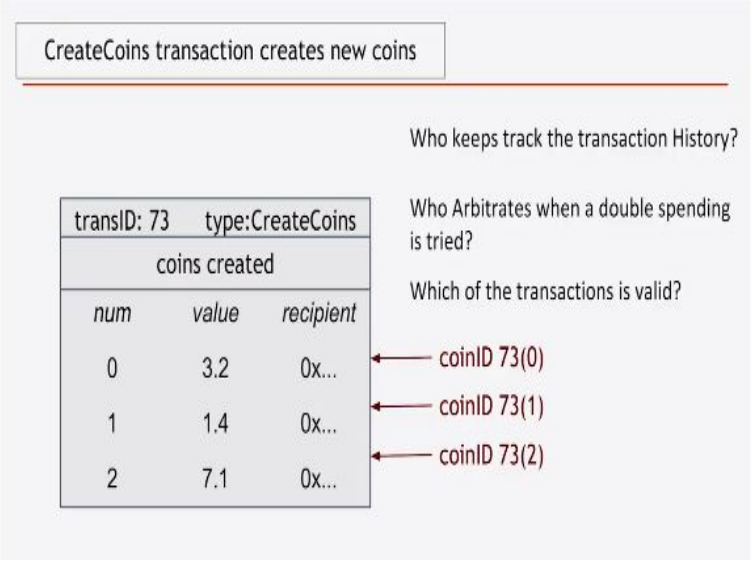
(Refer Slide Time: 59:37)



So what we, what Arvind did is that saying that we will create a history of all the transactions. So earlier when we were giving the coin to somebody, we were not giving any number to it, right. So we were just connecting them through this links. Now we say that we will have various transactions and each transaction will basically spend one coin, but we need to know where that coin came from, from which transaction.

So we have to go and see the next transaction and finally, there will be a transaction when the coin was created, right. So this history has to be kept.

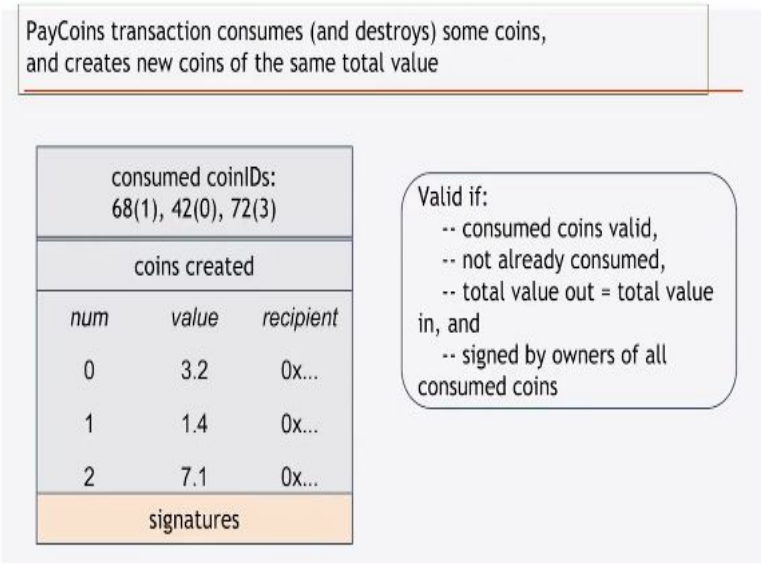
(Refer Slide Time: 1:00:17)



And this history now the data structure in which you keep the history and all that stuff is a matter of details like, but the main question is, who keeps track of the transaction history? Like who tells me that somebody is trying to do a double spending. And then if I have done a double spending to given the coin to two people, who decides who actually gets the money and who is the one who is not going to get the money.

And so which means that somebody has to tell me which one is a valid transaction, which one is not valid transaction.

(Refer Slide Time: 1:00:54)



So concept of validity here would be that the coin that has been used in the transaction is a valid transaction. Also, it is not already consumed, and the total value out of this transaction should be total value in. So if I am have multiple coins from multiple sources, I can use that in one transaction because not everything has the value of one coin. So I can get three different coins to buy a thing worth of three coins.

But then the total value in and value out should be the same. And then it should be signed by the owners of all the consumed coins, right. So this is the matter of validity. So that history that you saw, has to be of valid transactions, right? So somebody has to check the validity of the transaction and somebody has to use the history to tell me whether double spending is being done.

(Refer Slide Time: 1:01:45)

Immutable coins

Coins can't be transferred, subdivided, or combined.

But: you can get the same effect as subdivision by using transactions

- create new transactions
- consume your coin
- pay out two new coins to yourself

So now how do you actually maintain the coins? Whether you make the coins immutable and all this is a detail so we will not worry about it at this point.

(Refer Slide Time: 1:01:56)

Trusted Third Party

Crucial question:

You become the central Trusted Party keeping track of transaction history, arbitrarting validity of transactions

Why should people trust you??

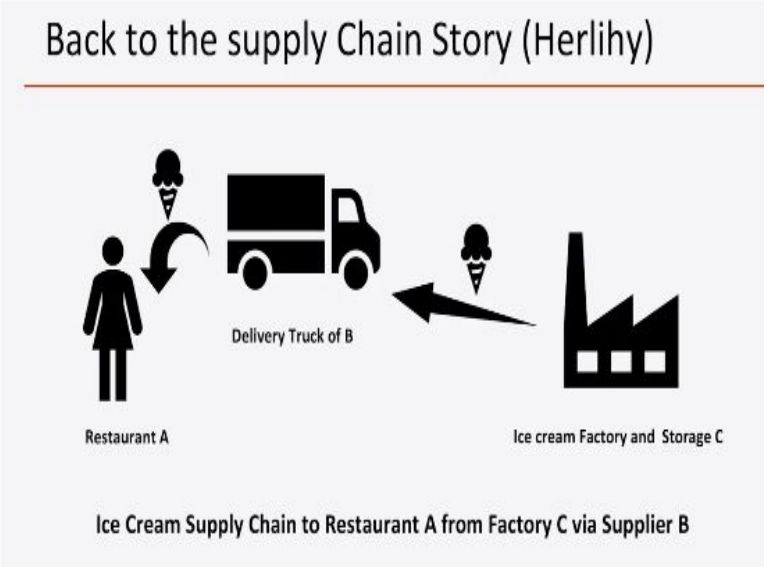
So the crucial question is who maintains the history and who validates the transactions, and who tells me that the this is invalid because there is a double spending. So one possible solution, of course, is having a trusted third party, right. So and once you have trusted third party, we are back to the same conventional problem that I have a trusted third party.

He maintains the ledger of transactions and he checks consults the ledger every time and coin is trying to be used in a transaction to ensure that the this is a valid

transaction. And but the question is, should we trust the trusted third party. The trusted third party might be a biased entity, and they might actually always arbitrate in favor of some people and arbitrate against some people.

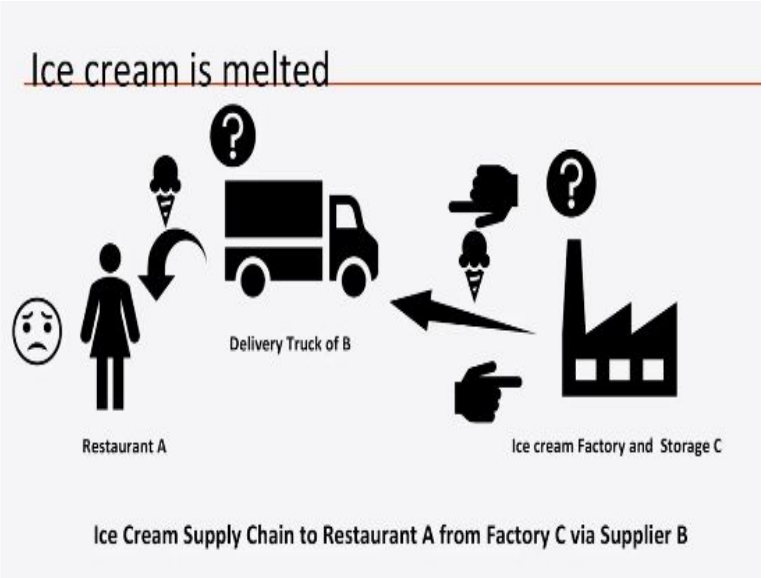
So it defeats the purpose of a distributed consensus mechanism and all that and that is where the blockchain based solution comes into play. So trusted third party is not our preferable solution because that solution already exist and being used today. But we want to remove the trusted third party and create a trust system in which we can transact trusting the system, not trusting a single entity. So that is the about that cryptocurrency issue.

(Refer Slide Time: 1:03:27)



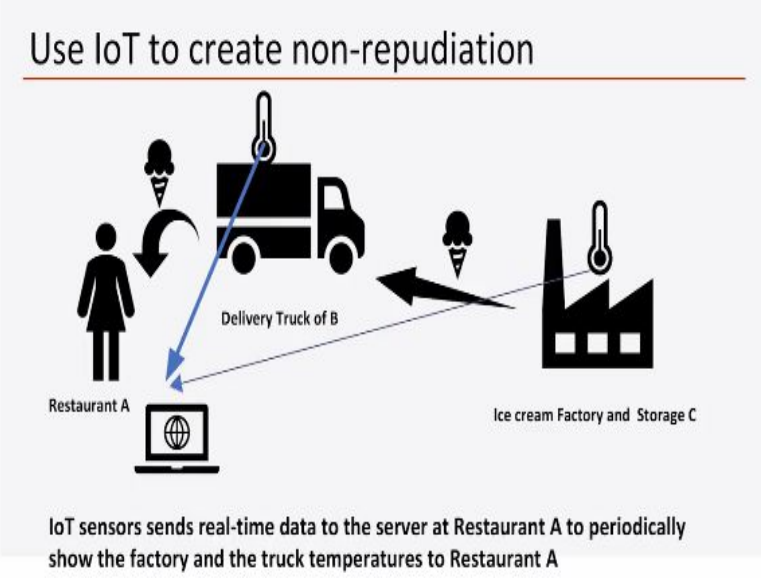
Now back to the supply chain story, right? So you have the Ice Cream Factory, you have the delivery truck, and you have the restaurant. And this question was that who is melting the ice cream, whether it is the factory when it supplies it to the truck, or the truck is actually keeping it at a very high temperature and the ice cream is melting.

(Refer Slide Time: 1:03:46)



So let us see what solutions, so the question is here is about, you know supply chain integrity. And when I get melted ice cream, I asked questions to both of them and both of them point fingers at each other.

(Refer Slide Time: 1:04:02)



So as I said earlier that I could put an IoT or Internet of Things type of sensor in each of these places in the factory and in the, in the truck. And I can say that send me your temperature every second or every 30 seconds. And I will keep track of your temperature where you store and I will actually decide who is responsible for this.

(Refer Slide Time: 1:04:34)

What can go wrong?

- IoT sensor data may be intercepted by a middle man and changed before it reaches the server (**data integrity**)
- IoT sensors may be stopped and old readings may be replayed (**replay attack**)
- What the server gets purportedly from factory C, may be manufactured by supplier B (**Authenticity**)
- If restaurant A claims that C's temperature reading shows that ice cream was melting in the storage, C can say that message you received is not from me – there was an MITM attack (**repudiation**)

- So restaurant A will not be able to pinpoint any one in the supply chain with full confidence!!

However, there are multiple problems that can occur. So let us talk about what can go wrong. First of all, if you do nothing other than just make these IoT devices send information, then the information is not encrypted and digitally signed, then somebody can do a man in the middle attack. So they can actually intercept the message, change it.

So maybe the company B will try to do that to make company C look bad by intercepting the temperature coming from company C and then replace it by a higher temperature. So data integrity could be a problem. And other thing is that if your temperature keeps going up because cooling system fail, you might fool the restaurant by doing a replay attack.

Which means you record the whole temperatures and then you replay it and send it repeatedly to the restaurant and the restaurant will think that your temperatures are fine. So that is a replay attack. Again data integrity issue. Then if the temperature that is coming from factory C is actually being manufactured by supplier B by replacing or jamming the signals from the factory C and pretending to be factory C and send the restaurant the bad information.

So there is an authenticity problem, right. When the data comes to a trust that this data is actually from factory C and not by sent by somebody else. And then if something

like this happens or not happens if factory C claims that this is what happened his his temperature was fine, but the temperature you are seeing is actually somebody else did an attack and and man in the middle attack and sent wrong temperature information.

And there is so he is repudiating what information has come from his supposedly from a sensor and that repudiation, so non repudiation is not a property. So restaurant A is again at a loss even though he spent money putting sensors with with communication capabilities, but he is not happy because he cannot decide what happened.

(Refer Slide Time: 1:06:47)

This can be solved without any blockchain issue, right. You can use message integrity proof by hashing or message authentication code. You can use digital signature for all IoT devices and then you have the authenticity and non-repudiation. You can use time stamping to stop replay attacks. And now factory A can get all the data with authentication with with privacy and with integrity, everything, fine, right.

(Refer Slide Time: 1:07:20)

Concurrency Issue

- A has other suppliers for other goods required for its business (multiple concurrent supply chains)
- B and C has multiple other consumers of their services
- So if there are N suppliers who are also consumers of some of these entities, we have an N^2 messaging problem

A offers that every one can look up their data from my server, so you can get linear number of messaging

But do you trust A as purveyors of your data?

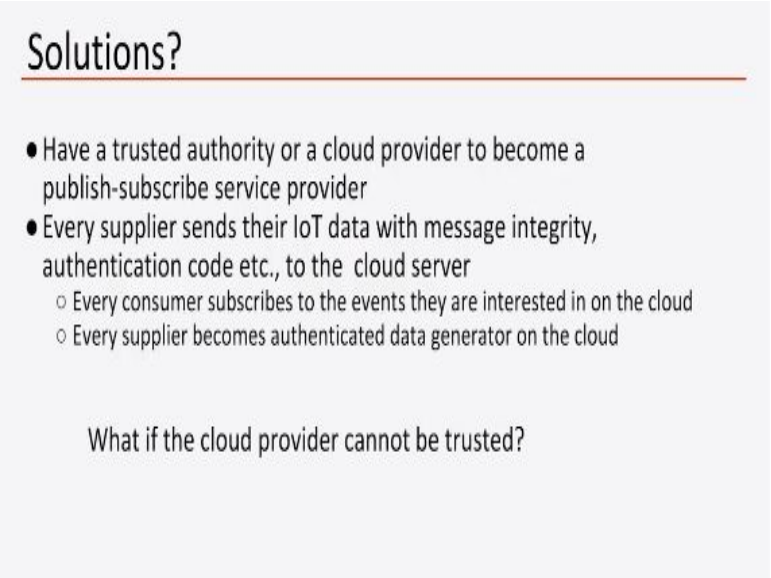
But the problem does not end here because supply chain problem has a concurrency problem. Because A is not the only consumer from supplier B or supplier C. Supplier B, C have other consumers. Similarly, restaurant A has other supply chains, right. So it gets meat from somewhere, it gets vegetables from somewhere. So there are many parties involved and if you want to check supply chain provenance for all your supply chains, then you have a n square problem.

Because every party should be able to send information to everybody else whatever the sensor readings are and the sensor could be temperature, it could be about the pH level, it could be about various things. And therefore, it is a huge amount of crisscross information. And that needs to be addressed because it is not going to scale. So restaurant A may say, okay, I have a very good nice data center.

So I will keep everybody's information. So you just only have to send to me and then I will give a nice interface to the information I collect. And everybody else, every other restaurant and every other supplier can login to it and look at the information and make themselves happy, right. But then why would you trust A to be the purveyor of your data. What if A wants to make some other entity look bad and tampered the data.

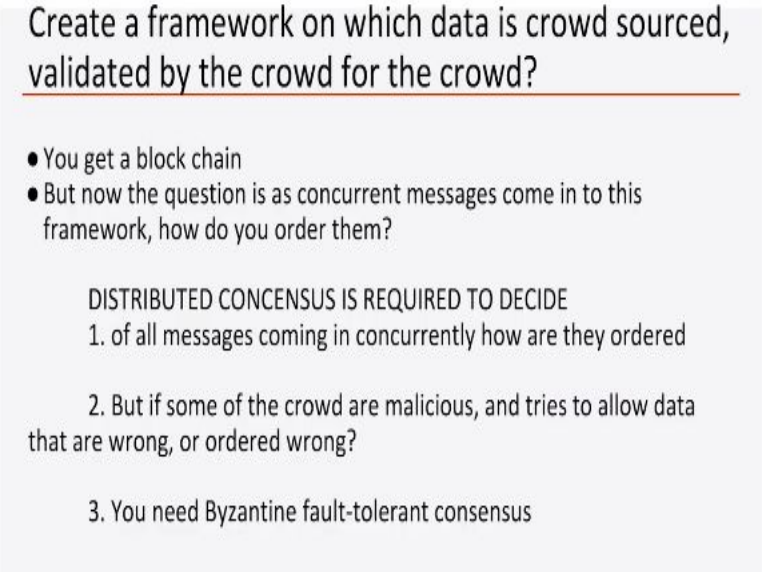
So when you when that entity, another entity looks up the data for that entity sensors, you might see wrong information, right? So you cannot trust a single entity to do all the job because then it becomes a trusted third party again.

(Refer Slide Time: 1:09:02)



So you can use a cloud provider and say instead of A, B, or C or anybody, the cloud provider collects all the information and most IoT vendors now is giving a solution based on this cloud based data collection, and then interface to the cloud to look at the browse the data and do analytics on the data. But what if the cloud provider cannot be trusted? That is a possibility. Right? Because again, a trusted third party.

(Refer Slide Time: 1:09:31)



So then, the question is, you can use blockchain because blockchain is not a trusted third party. Blockchain is create trust computationally by using a distributed consensus mechanism. So all the information that comes to the blockchain is seen by all the players in the blockchain and all the players have to somehow agree that the information is correct. And once the information put there, nobody should be able to tamper with it. Right?

So blockchain is actually a framework that can be actually a very good use in this supply chain case that we just described. So we will see later that various kinds of blockchain solutions to this kind of problems, not necessarily will take this problem, because this problem is probably a too simple a problem to discuss later.

But you get the idea that here, important part is that even with all the cryptographic measures like digital signatures, and you know encryption and message authentication code, you still have a scalability problem and concurrency problem that needs to be solved. And the one simple solution would be to have a trusted third party to collect all the data and keep it.

But trusted third party is not something we want because it again brings us back to the old traditional mechanism where you have to somehow put your trust on a trusted third party. Instead, if you can create a computational platform in which trust is by virtue of the computational process, then it is much better solution and more fair solution. So so that is what we would like to do. So we are at the end of our first lecture.

(Refer Slide Time: 1:11:20)

Conclusion of the First Lecture

- Blockchain is about
  - Distributed Record Keeping
  - Trust Model varies – but usually single point of trust is not good
  - Based on Trust Model –
    - Permissioned Blockchain
    - Non-permissioned or public block chain
    - Also, private blockchain
  - Data integrity (No one has tampered with the data after its creation)
  - Authenticated Transactions or event logging
  - Strong Cryptographic Application
- Blockchain is certainly not ONLY
  - Cryptocurrency
  - In this course, cryptocurrency will be avoided

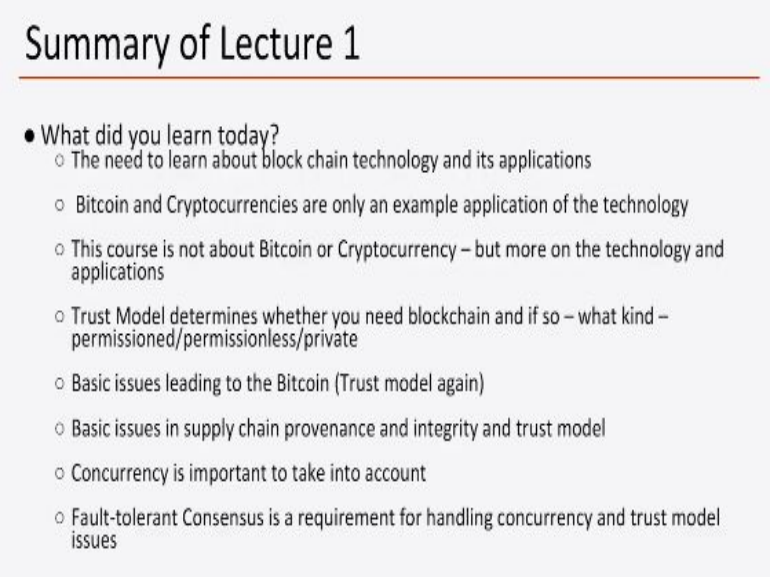
So in the first lecture, what we learnt is that blockchain is about distributed record keeping. Trust model varies, but having a single point of trust is never good. And based on the trust model, you may have to use blockchain. But depending on who you trust and how much you trust, you have multiple different types of blockchain. So we will see permission blockchain non permission or public blockchain and also private blockchain.

So these are the some of the different alternatives and based on how much you trust which party and and how much you know about the players who will be actually participating in the consensus process, in the validation of the data process, then you have to choose the different types of blockchain and we try to learn how those things are done, or what are the different considerations that go into choosing these kind of alternatives.

Data integrity, we talked a lot about. No one tampered with the data, after its creation is very important. And that is where blockchain comes into play and authenticated transactions and event logging is something very important if you want to know not necessarily the data itself, but metadata, that is the data about who access the data and who modified the data, who added the data, that kind of information.

And strong cryptographic applications. Blockchain is a strong use of cryptography. And blockchain is not certainly only about cryptocurrency and in this course, we will not be focusing on cryptocurrency. But we will talk about some of the technology of bitcoin and ethereum and all that, because we need to understand the underlying technology.

(Refer Slide Time: 1:13:14)



So summary of lecture 1. So what did you learn? The need to learn about blockchain technology and applications is what we emphasized on today because of the various applications we talked about. Bitcoin and cryptocurrency are only examples of this technology. But these technologies have much broader usage and much more potential. Trust model depends determines what kind of blockchain you are going to use or if you at all need block, will blockchain solve your problem?

Because that is something also one has to take away from this course because many times we I get request that can you help us to do blockchain for our application. And when you actually investigate closely, their trust assumptions and their their operating environment and you find that under their trust assumptions and everything blockchain is not required or it will be an overkill right.

So you have to also get the have to have the critical understanding of when blockchain can be of help and have value. And also what kind of blockchain is good

for that particular application. Basic issues that led to the creation of bitcoin about the not trusting the banks and the or a trusted third party or a central authority and supply chain provenance and integrity and trust model concurrency issues that arise when there are many players.

And maybe in that case, central data collector like a cloud may be or maybe not a good solution. And fault tolerant consensus is required in working of the blockchain consensus mechanism and workout the trust model issues. So this is the introduction that we have gone through today. In the next lecture, we will talk about some basics of cryptography because you will you already heard a lot about private key public key and hashing and message authentication code in today's talk.

And many of you know what they are, many of you do not. So in the next lecture we will actually familiarize you with all those terms and what they mean and how they are calculated. So that will be, then the introduction to this course, will be complete once we have lecture 2 on that on crypto concepts. Thank you.

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

Lecture - 02

So welcome to lecture 2 of Blockchain Technology and Applications on NPTEL. So today's topic is more on the cryptography and the basic cryptographic concepts that are being used in the construction of blockchains and protecting data integrity of data in blockchain.

(Refer Slide Time: 00:42)

So the material in this lecture are from various lectures and courses and writings of various colleagues around the world. So I should acknowledge them first and then we will move on to the topics.

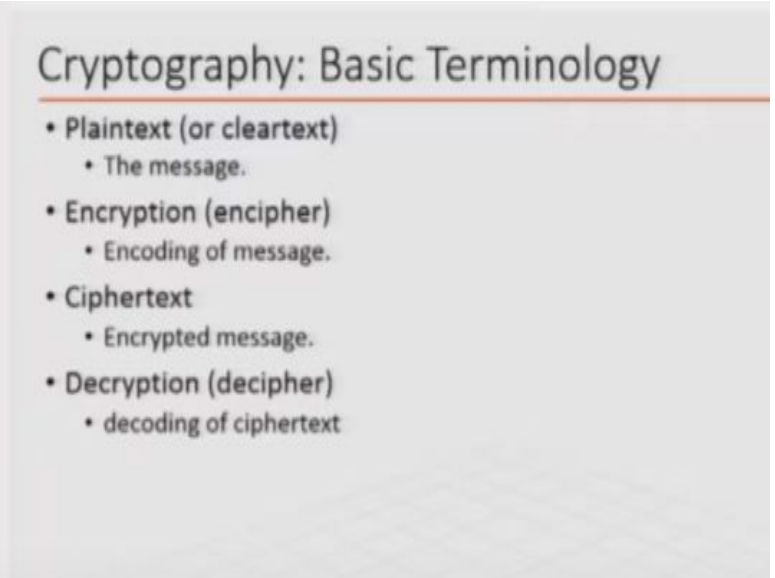
(Refer Slide Time: 00:55)

So today's topics are first of all, the basic of cryptography and this is not a cryptography course. So therefore we will only focus on the very basic essentials that are being used for blockchain technology. And one of the most important part of constructing a blockchain is hash functions. And hash functions are suitable because of certain properties of hash function that determines what function could actually be used as a hash function.

So we will talk about that in detail. We will talk a little bit about public key cryptosystem because in most blockchain, the public key of an individual is used as his address, address of an account. So public key cryptosystem understanding how it works is very important. Another thing that is very important in blockchain is digital signatures because all transactions must be signed digitally by the appropriate person or appropriate account.

And then we will talk a little bit about hash puzzles and hash pointers. And finally, we will talk about Merkle data structures, which is the data structure that is most commonly used in storing large amounts of data in blockchain. So basic concepts of cryptography.

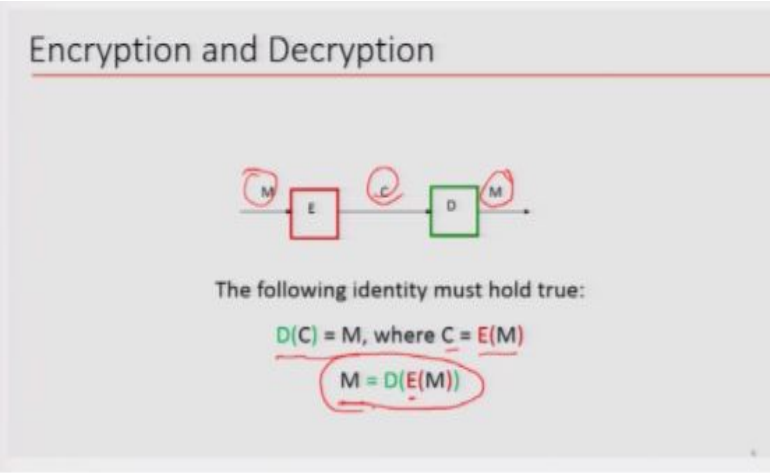
(Refer Slide Time: 02:22)



So some of the terms that you will hear a lot when you talk about cryptography are plaintext, or cleartext. That is the message that you want to encrypt. So it is plaintext, because if you read it, you get it, right. The second thing is encryption or encipher. That is an action that you apply on the cleartext or the plaintext and you encode it in a form so that only people who have a very specific knowledge can actually recover the original plaintext or original message, others will see garbage, right?

So that is the idea of encryption. This is a very old idea, you know you know we have, we know that Julius Caesar used Caesar cipher. So this is not something that is very modern. But in modern day, the technology, and the algorithms for encryption or enciphering has changed drastically from the old days. Then after the encryption, what we get, we call them ciphertext. It is basically the encrypted message. And in order to read the message, you have to do a decryption or decipher. And this basically converts the ciphertext to plaintext.

(Refer Slide Time: 03:46)

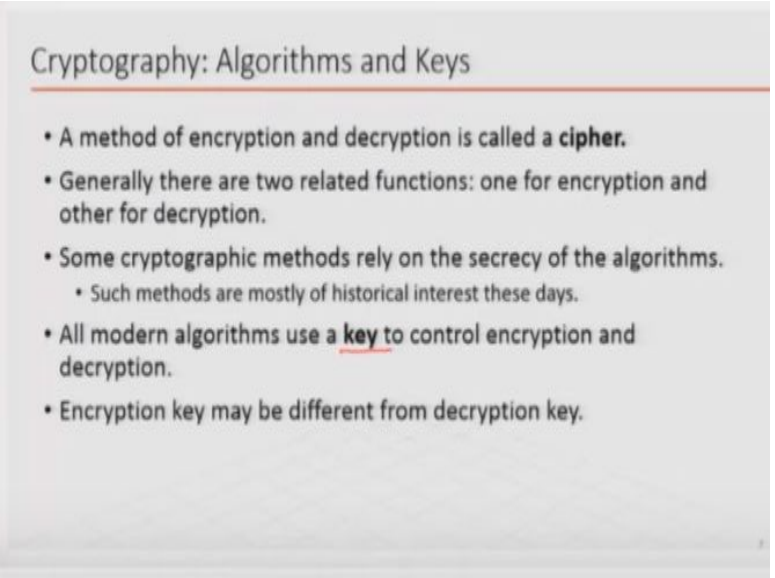


So now if you look at this as some kind of a data flow, then **M** is your message or plaintext here. And this plaintext goes into an encryption algorithm or here it is shown as an encryption box. It could be hardware, it could be software and then what comes out is the ciphertext. The ciphertext is the encrypted version of the original message.

And then you want to read it, then you have to pass it through the decryption or deciphering algorithm or the hardware and then comes out the cleartext or the original message. So the following identity must hold. So if **D applied to C is M**, and **C is E applied to M**. Then **M** must be recoverable by a **composition of D and E**.

That means, if you apply E first on the message, you get the ciphertext and then on the ciphertext, if you apply D, then you should get back the plaintext. So that is the idea, that is the property of all encryption systems.

(Refer Slide Time: 05:01)



So in cryptography, as we said that the method of encryption and decryption is called a cipher. So generally there are two related functions, as we showed in the figure, that there is one for encryption and one for decryption. And in the old days, the how you encrypted and how you decrypt it were kept secret, right? That is the only secret that stood in the way of somebody an eavesdropper.

Somebody who is trying to capture the message on its way and trying to understand it, it is secret to them, and therefore, they cannot do it, unless somehow they figure out what the algorithm is. And this is what happened in the 1940s when the Germans used this Enigma code to instruct their naval ships and U-boats. And this messages were going encrypted.

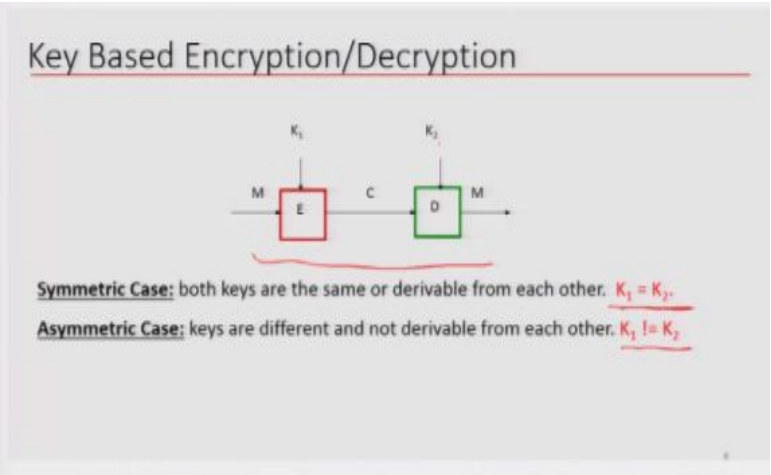
And then the Britishers, which was a team led by Alan Turing, actually constructed a method and device that would reverse the process. And without knowing the algorithm that the Germans were using, they had to actually come up with the algorithm that is being used. And then they had to figure out how to reverse it. And that is how they actually encrypted the Enigma code.

But nowadays, we do not keep the algorithm secret, because keeping an algorithm secret is quite difficult. So we have to use some other method to keep the secrecy and that is where the idea of a key comes in. So a cryptography key is basically the secret

that is an ingredient into the encryption process. And the key is also an ingredient in the decryption process. So the encryption key and the decryption key, maybe same or maybe different.

And based on whether they are same or they are different we have two types of encryption or decryption method, which are called the symmetric encryption or asymmetric encryption.

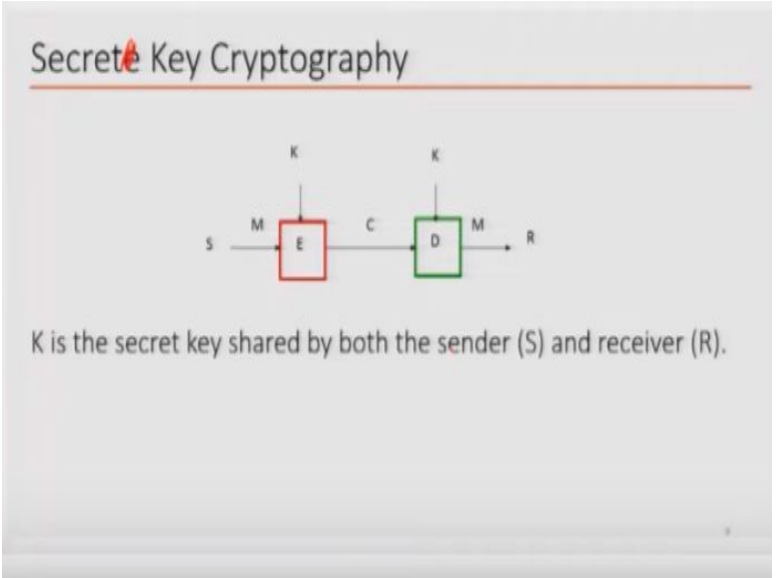
(Refer Slide Time: 07:13)



Now you see that the picture I showed earlier, the second inputs to this boxes were not there. So now we have the encryption in encryption box which takes a message and a key, in this case **K 1**. And then outcomes, the ciphertext. And then when you put that ciphertext to the decryption algorithm, then you need another key or maybe the same key depending on whether it is a symmetric case or asymmetric case.

And then you get the **M**. Now symmetric basically means that the two keys **K 1** and **K 2** are the same whereas or they are derivable from each other. And in case the two keys are separate and not derivable from each other, then we say that it is an asymmetric algorithm. And we will see that both of them have their own importance in the whole literature of cryptography or the practice of cryptography. And we will talk about that.

(Refer Slide Time: 08:22)



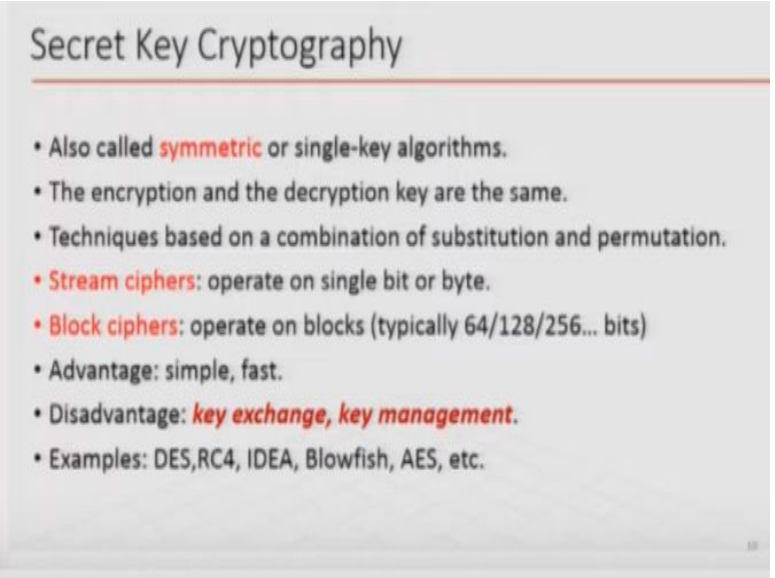
So this is also called a secret key cryptography. So secret key cryptography means that the key is has to be known and kept completely secret from outsiders. So only the person encrypting it should know the key and the person who is supposed to decrypt it should know the key. So that means this key is shared between the encrypted and decrypted. And this sharing is a big problem because how do you let the decrypted know the secret key?

So one possibility is that you do it out of band which means that you use some other means like a, like a telephonic call or some other kinds of method in which you send the key to the receiver, and then you send the message and then the receiver will use the key. But if you send the key with the message, then anybody who is listening on the communication channel will read the ciphertext, but then also read the key.

And therefore, they will use the key to decipher because algorithms are not secret, only the keys are secret. So therefore, there has to be a way in which the key is shared between the two parties. And that is one of the problems that the symmetric key cryptography suffers from. And since before you share a key, you cannot encrypt the key, right? Because then there is a problem that when you have the key, then you can decipher it.

But if you do not have the key, then there has to be a way to share it. And most of the time, this has to done out of band. And which means that there has to be some other way of communicating between the sender and the receiver. So this is one of the problem of symmetric key.

(Refer Slide Time: 10:10)



So this symmetric key is also called single key algorithm because you have only one key shared between the receiver and sender. And obviously, two keys are the same. And most of the time this encryption algorithms are also pretty simple. They are simple, but they are tedious. So you take the information that you want to encrypt.

And with the key using the key, you decide on certain substitutions and the part of the message or permutations of the bits or symbols of the message and you do this for many, many rounds. And how you do it depends on the key, the individual bits of the key. And therefore, when you want to decrypt it, you have to reverse the permutations and reverse the substitutions in order to get this message back.

And since you have the key, you know what the substitution and permutation sequences were, and by how much the permutations happened, how by how much the substitutions happened. So similarly, you will do the opposite in the other side. One good thing about this is this substitution and permutation are pretty easy to do in hardware.

Because you are basically taking bits and pushing them through shufflers and parameters and all that stuff. And you do this many, many times. They are called rounds. And therefore, the secret key cryptography has the advantage of being very efficient and fast when you have a hardware support for doing cryptography. So there are chips that will do this for you in a very, very fast turnaround time.

So now how you encrypt with the help the key is based on whether you are doing it continuously as you see bits of the message and you do something with the bits or you divide the message into blocks, chop the message into multiple smaller messages, and then you apply the key as a whole on each of this chopped portions. So those are the depending on how you do it, whether you want to do it on the stream of bits or on the blocks of bits.

These are called block ciphers and stream ciphers and block ciphers usually divide the message into chunks of like 64 or 128 or 256 or 512 bit blocks. So as I said that the symmetric cryptography is very simple and very, very fast. So disadvantage, as I said, is that how do you exchange the key between the sender and the receiver. And if you want to keep the key in some kind of a, you know database or something, there is a possibility of the key being stolen or read by somebody.

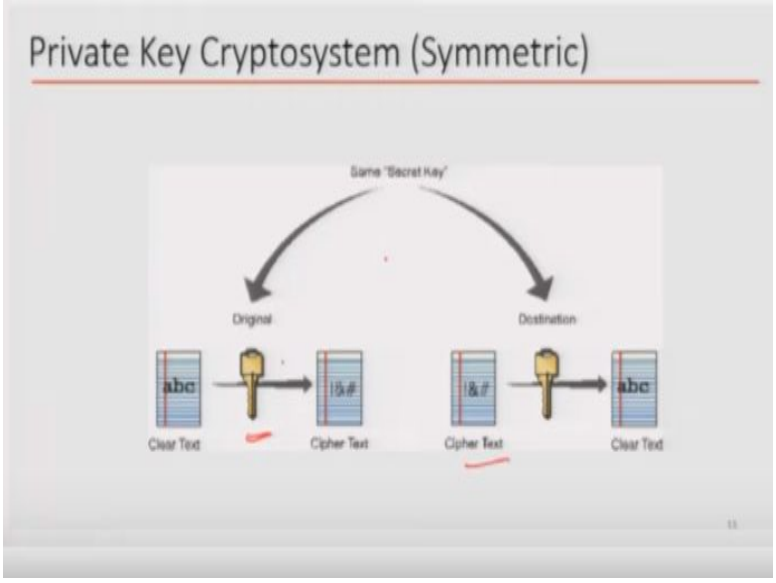
Because if you keep it in an encrypted form, then the you have to first give the everybody some kind of a key to decrypt them, right. So you have to keep it in plaintext, that is not safe. And therefore, key exchange and key management is a big problem in symmetric key cryptography. And that is where the asymmetric cryptography comes to help because the asymmetric cryptography is used to actually do the key exchange.

And once you have done the key exchange, you can apply symmetric key cryptography. Now you might ask, why should not I then only use asymmetric cryptography? The problem is asymmetric cryptography is not as efficient and it is time consuming. Therefore, you only use it very, very sparingly. And in most cases, it

is used for key exchange and after the key exchange has happened you can then apply symmetric cryptography on very large amount of data, because it is fast.

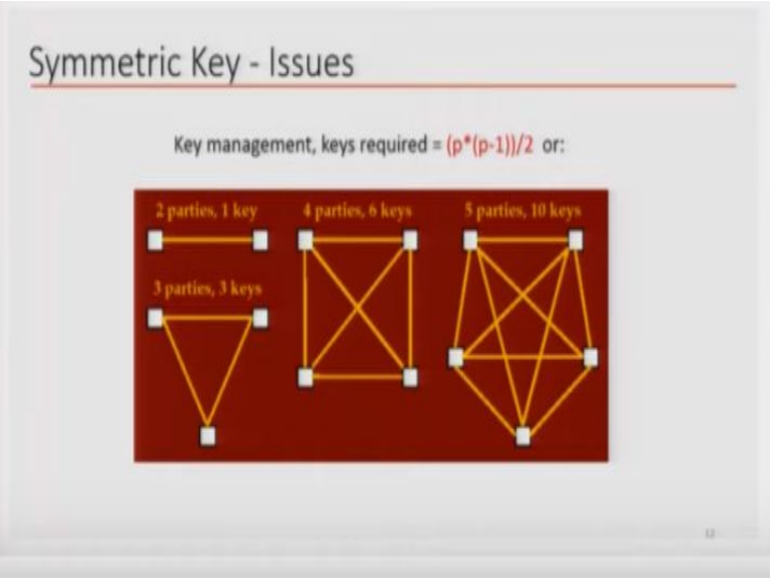
So examples of symmetric key cryptography are DES, like Data Encryption Standard RC4, IDEA and Blowfish, and most common nowadays is the AES, right. So so AES is the current most common standard and there are variants of AES, which are also used.

(Refer Slide Time: 14:31)



So pictorially if you look at this, the private key cryptosystem has the cleartext to ciphertext with a key and then the same key, the same key can be used for, from ciphertext to the cleartext. The only issue is how do you share the key.

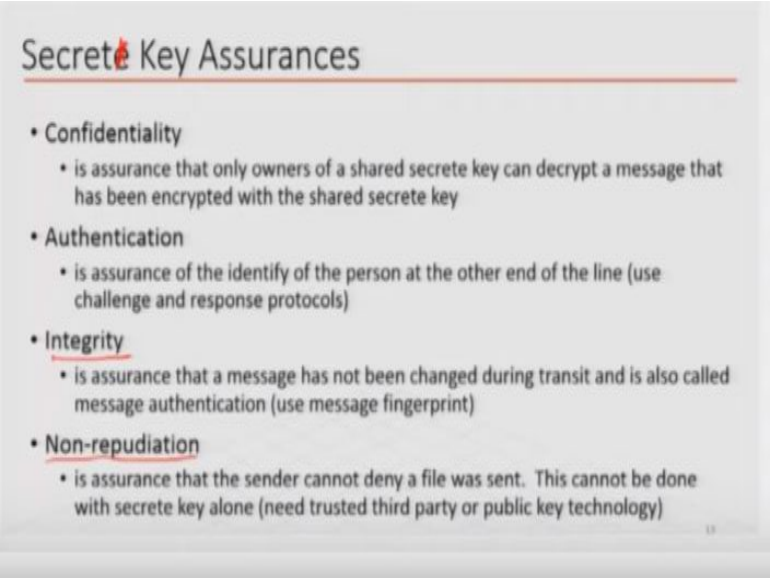
(Refer Slide Time: 14:54)



The other problem is that if you have 2 parties, then you have 1 key but if you have 3 parties, then you need, you know **3C2**, which is 3 keys. If you have 4 parties **4C2** six keys, and if you have 5 parties, then you have, you know **5C2**, which is 10 keys. So this way, the number of keys between parties will also increase very, very quickly in quadratic speed.

And that is another problem that you know you have to maintain so many keys for every pair of communicators. So that is another issue with the symmetric key.

(Refer Slide Time: 15:33)



So let us see what are the properties of secret key or symmetric key algorithms. First of all, it assures confidentiality. So which means that if you have already exchanged

the key, and nobody else has the key, then when you send the ciphertext on the way, if somebody reads your ciphertext by tapping on to your network, they would not be able to figure out what what is being sent because unless they have the key they cannot decipher.

So confidentiality is one of the primary reason why we do cryptography. So the symmetric key cryptography basically achieves that privacy or confidentiality of information, provided the key is kept secret. Also if the key is kept secret, then authentication. That is, if you receive a message from somebody, you want to know that it is that person who I suppose to send you the message is indeed sending you the message.

And somebody is not pretending to be him or her and sending you the message. Now if you have done the key exchange beforehand with the authentic person, then when you receive the message and you apply your key for decryption, and you could and you can decrypt, then that means the anybody who has sent this message must have the key.

Now assuming that the key has not been stolen from the sender who you actually exchange the key with, then the message has to be coming from an authentic source. So authenticity is also proven, provided the key is kept really secret between the two parties. Message integrity, integrity means that the message or the data has not been tampered with, nobody has changed the data while the data was in transit.

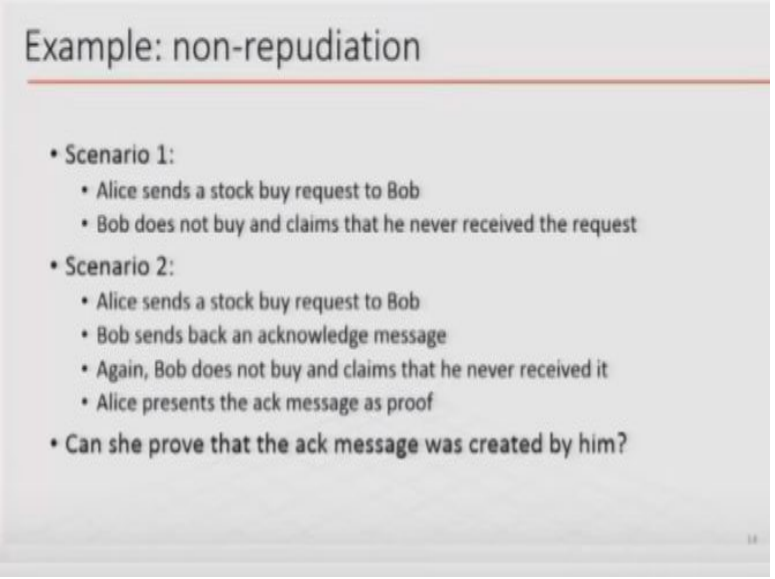
So that is also assured because if somebody replace the message, and then he has to also encrypt the message with the key. Otherwise, you would not be able to decipher it. When you try to decipher it with a key and it was actually encrypted with a different key then you will get back garbage. So therefore if the if you can decrypt the message, then it is actually showing that the message could not have been tampered with.

So message integrity is also proven if you can decipher the message. And non-repudiation. Non repudiation means that the sender cannot later say, oh no, I did not send this message. Because if the sender has used the key, and he is the one who knows the key and nobody else, then he has to have sent it. Otherwise, there are two possibilities. One, the key is stolen or he is lying.

Assuming the key is not stolen, then it is a non-repudiable. That is that the sender cannot admit to have sent the message, he cannot deny. So now let us look at non repudiation a little more closely by playing out a scenario. So let us say and in cryptography literature, you will always see these names Alice, Bob, Eve. So usually Alice and Bob are the sender receivers, and Eve is an eavesdropper, right.

So who actually tries to capture the message while the message is on transit and tries to decipher. So Eve is usually the attacker, or threat.

(Refer Slide Time: 19:20)



So in this case, suppose Alice sends a stock buy request to Bob, and Bob does not buy and claims to have never received the message. So at the end, Alice would suffer some loss because the stock might have gone up in price. And then Alice says to Bob that I was supposed to have bought this when the price was low, and I sent you message but you did not buy it. So I have suffered loss. Bob can say that no, no, I never got the message so I never bought that stock.

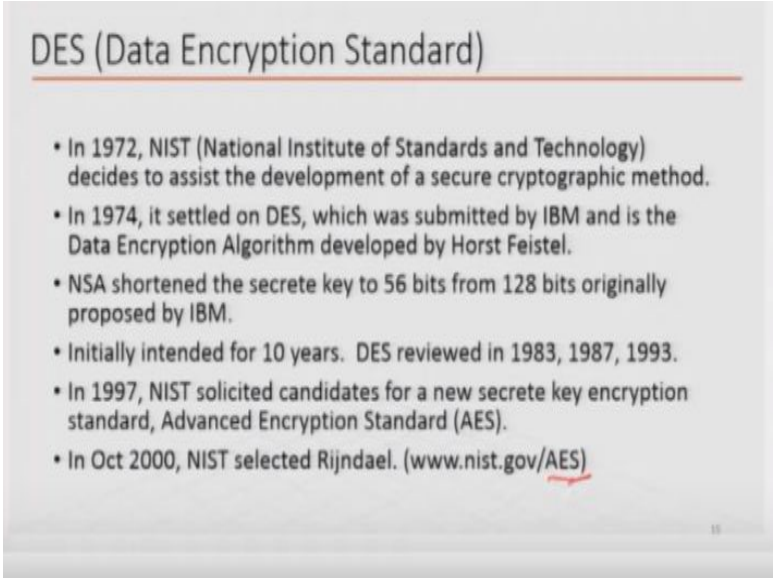
So he can deny to have received the message. In scenario 2, suppose Alice sends a stock buy request to Bob. And Bob sends back an acknowledgement that yeah, I am going to do it, I received your request. And then again Bob fails to buy it. And then later when Alice comes and says I have suffered loss for you, because of you. Bob can say that no, no, I did not get the message.

Then Alice shows him that look, you sent me an acknowledgement message. And therefore, you must have received the message. Now in case they are not using encryption, or any kind of method of non-repudiable communication, then Bob can say that well, it looks like a message from me. It has on its header that it is from my email address, but somebody fabricated it.

And then without the use of cryptography for Alice, there is no way that she can prove that the acknowledgement message was indeed from Bob. Maybe somebody, like an Eve might have captured Alice's request in the in the middle of the transit, and then block the message to receive go to Bob. And she faked a message from Bob to Alice that okay fine, I am going to buy it, but she does not do it. So Bob is not the culprit.

So in this case, Bob has repudiability. But in case of, in case the the whole thing was done cryptographically, then Bob could not have done repudiation unless he claim that my key was stolen, which is another issue outside the scope. So in a court of law, under the circumstances we described, Alice cannot prove that the acknowledgement message was indeed from Bob. So that is about non repudiation.

(Refer Slide Time: 21:53)

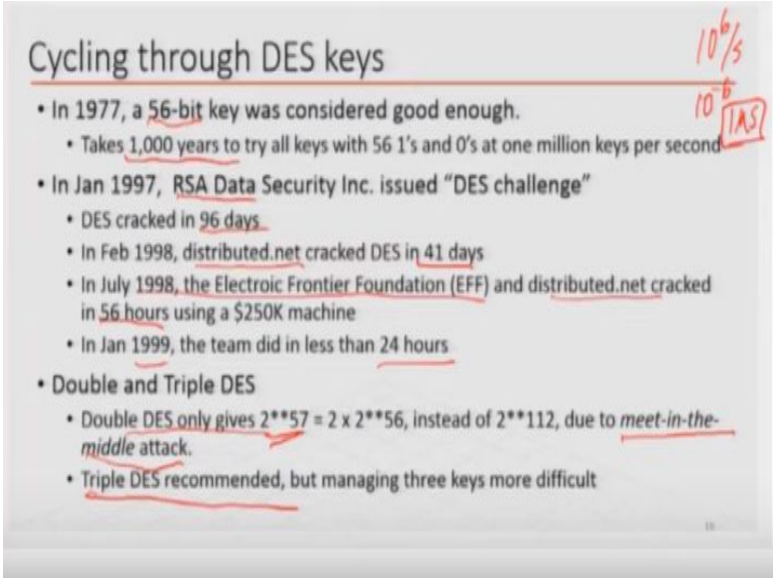


So now let us look at what are the different symmetric key algorithms what how they came about. So the National Institute of Standards and Technology in the US in 1972, decided that cryptography is becoming important. More people are communicating through various networkings. So so let us define cryptography. So what NIST does is that they opens challenge for anybody to respond to.

So IBM sent one of the entries to the challenge, and it was called the DES and turns out that was accepted by the community as the NIST as the standard. And although IBM initially said in their proposal that it should be a 128 bit key, but NSA, the National Security Agency of the US, they decided that 56 bits is enough. So a 56 bit key will protect the information. And they said we will do this for 10 years, then we will revise the standard.

It was reviewed several times until 1993. And then they realized that we need a new key, secret key algorithm. And that time again they solicited response from people and eventually in 2000, they selected Rijndael, which is actually was originated in Belgium. And that is the standard called AES, Advanced Encryption Standard. And that is what is in use mostly these days.

(Refer Slide Time: 23:35)



Now what is the reason why DES, you know failed to be the standard for too long. Remember that 56 bit keys means that there are 2 to the 56 possible keys. Now if I am an attacker, and I do not know the key, so I will try all possible combinations 2 to the like 000 like 56 zeroes to 56 ones and any any other combinations in the middle, so there are 2 to the 56 such possible combinations. And I will try those on the message.

And if one of them unlocks the message, then I would know that is the key. However, with 1970s computing power it and even if they try out 1 million keys per second, even then, which is actually at in those times was impossible, 1 million is 10 to the 6 so per second would mean that each key can be tried in 10 to the -6 that is 1 microsecond, which was not even possible in those days.

But even if it was possible, then they calculated that it will take somebody thousand years in those computers with the you know assuming that the computer is pretty fast and it can do one attempt in one microsecond, it would take 1000 years. So then RSA the company, another cyber security and encryption company, they issued a DES challenge in 97.

And very quickly, some of the people who took the challenge they cracked DES in 96 days as in 97 the computing was lot more fast and lot of more memory was available plus lot of parallel computing was available. In 98, the another player in the

challenge, they reduced it to 41 days. And then in 98, the Electronic Frontier Foundation and the distributed.net, they reduced it to 56 hours.

And in 99, they did it in less than 24 hours. So at that point, they decided that this is not safe, because attackers could use very you know especially if the attacker is from a big nation, the government is behind it, then they can use it to crack the keys very very fast and therefore, they decided that this is not safe. So one possibility is you double the DES right? So you say that first you apply DES once, and then use another key to do the DES on the encrypted one again.

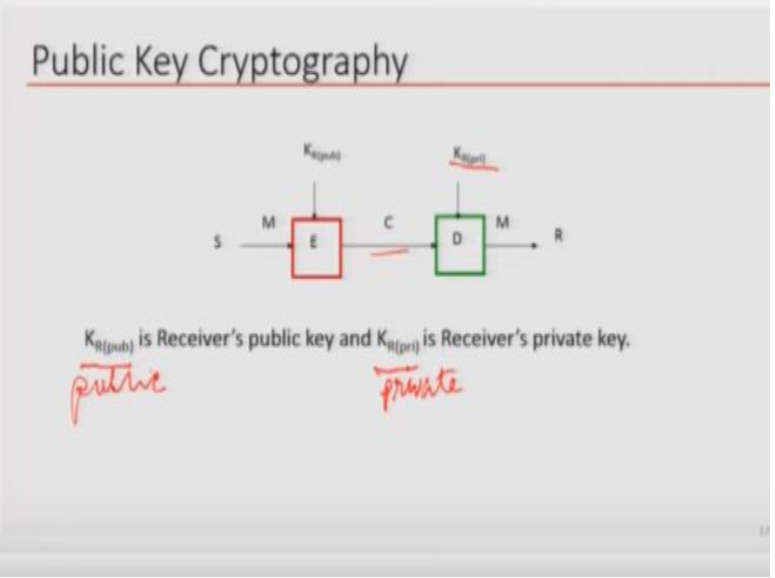
So if somebody has to unencrypt it, unencrypt it, then they have to do the opposite that is they have to first find a key for the for the second encryption and then apply it and then they have to do another time with the second key and the first key and then they will be able to decipher it. So that would mean that they have to try $\mathbf{2^{56}*2^{56} = 2^{112}}$ keys.

But it turns out that there is a meet in the middle attack by which you try from both sides. And therefore, it turns out the double DES is equivalent to having a strength of 57 bits rather than 112 bits. So the NIST recommended that you have to use triple DES which is safer. Now triple DES means that you have to have 3 keys for each pair of communicators, right. So three keys for each pair of communicators would basically make this key management issue even more problematic than before.

So therefore, public key cryptography is something that is almost essential in actually doing cryptography today. In practice, for example, when you communicate over the web, and you do ecommerce on the web, you use HTTPS, which uses SSL, secure socket layer and there the key exchange happens through public key cryptography. Once the key exchange happens it is called a session key.

And then that session key is used as a symmetric key between the two parties. And public key cryptography in the beginning, allows them to establish that secret key between the two parties.

(Refer Slide Time: 28:19)



So what is a public key cryptography as we said before? In public key cryptography, there are two keys. One is kept secret. The other one is kept public or known to everybody. And this pair of keys here we are calling it $\mathbf{K_{R(pub)}}$(public key) and $\mathbf{K_{R(pri)}}$(private key). So what you do is suppose you want to send me a message and I tell you my public key.

I can tell anybody my public key, and that time, what you do is you encrypt with that public key, transcend me the ciphertext. Since I have the private key and nobody else has a private key corresponding to this public key, only I can decipher the message. And I do not tell this key to anybody, I kept keep it very closely secret. So therefore, the key, I have two key or I can say it is a two part a key system.

One part, the public part, I can announce to the world that if you want to send message to me, then send me encrypted with this key. And the way it is designed that only only person who can actually unencrypt the message or decipher the message is one possessing the private key. Therefore, it has to be only me who can read it. So that is the idea of public key cryptography, right.

(Refer Slide Time: 29:49)

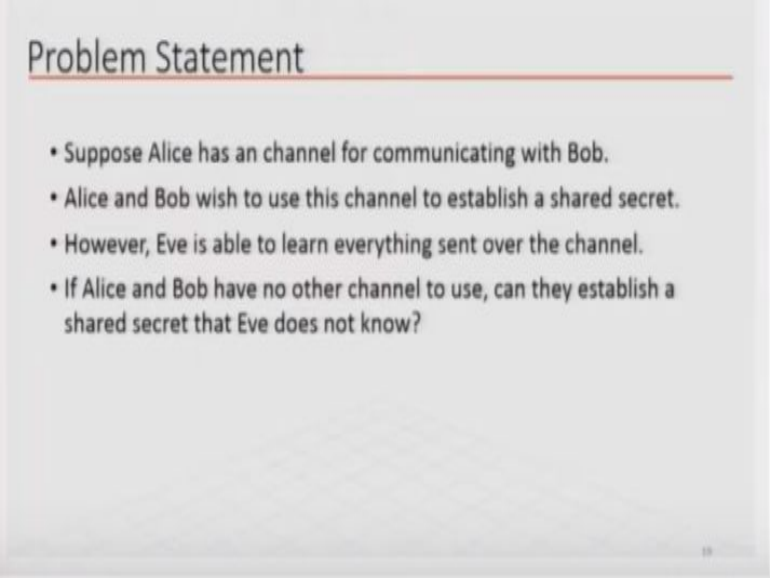
So so now the question is, how is it used to establish a shared secret, because as I said that public key cryptography is usually based on a very complex mathematical problem compared to secret key cryptography or symmetric key cryptography where bit shuffling and bit permutation, this kind of stuff is used. So it is very fast to implement.

Whereas in this case, normally what we do is we try to solve a very complex mathematical problem, rather than a bit shuffling kind of thing, simple things. Therefore, it is not very fast. So I have to eventually use a symmetric key cryptography. But to establish the secret key between the two parties, I need to use public key cryptography. So that is what public key cryptography is normally used for.

So I am sending a message over the Internet to my friend. Here in the internet, there are lurking behind many of the computers, there are people who can capture my message. And then if they know the key, then they will decipher it. So I have to somehow make the system such that the symmetric key that I am going to use to communicate very large amount of messaging or file uploads and all that I will not let anybody else know the key. So public key cryptography helps me by the following.

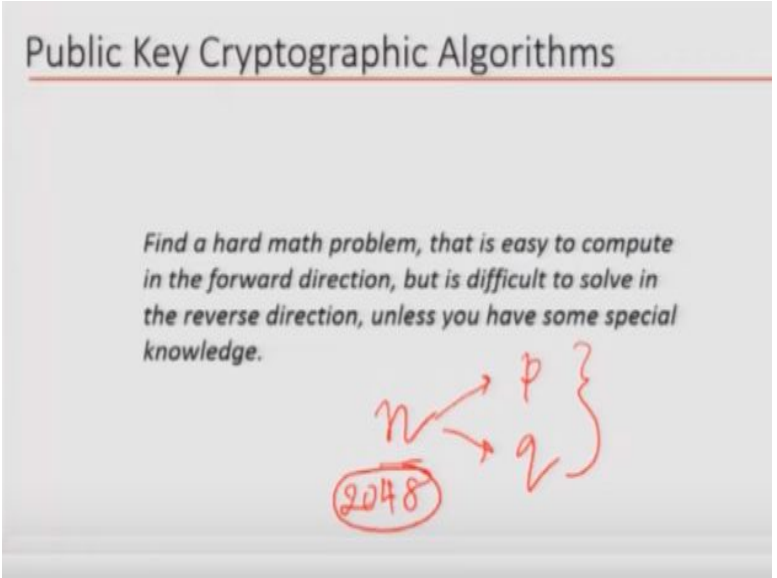
If you use public key cryptography, then you never send the secret key over the network. So what you do is you establish the secret key by using public key cryptography without having to ever sent the key on the network. So everybody creates the key on their own computer. And because of the mathematical process that is followed, the key this party establishes and the key the other party establishes becomes the same. So that becomes a shared secret without having gone through the network.

(Refer Slide Time: 31:48)



So the problem statement here would be that Alice has a channel for communicating with Bob and they want to use this channel to establish a shared secret. However, Eve is able to learn anything that is sent over the channel. So if Alice and Bob has no other channel to use, how can they establish a shared secret?

(Refer Slide Time: 32:07)

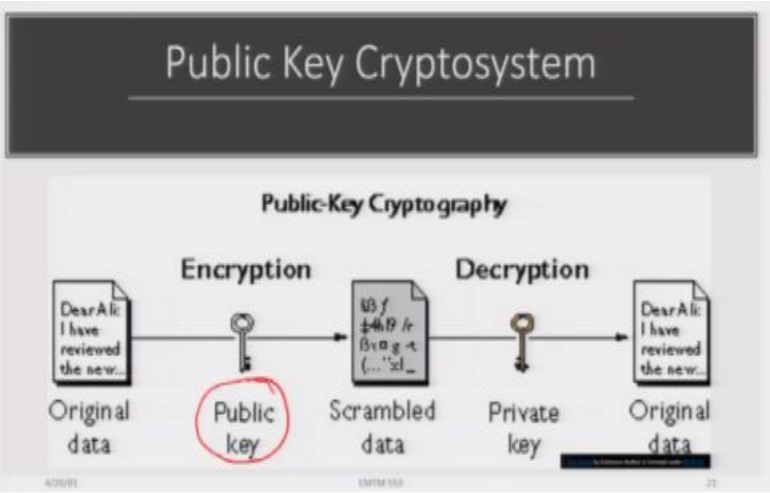


So public key cryptography is as I said is based on a hard mathematical problem. So for example, RSA is dependent on the fact that if I give you a very large integer n and the n can be factored into two prime numbers, p and q, finding those p and q to factor n is a very hard problem. So it is called the integer factorization problem. So figuring that this problem is not very fast and if n is a very, very large number like 2048 bits.

This is a 2048 bit number. So it is a very large number, let us say. And then you have to know what are the two prime numbers that produced this number by being multiplied, it is a hard problem. And this kind of hard problems are used for public key cryptography. So so these are also called one way functions because multiplying p and q to produce n is easy.

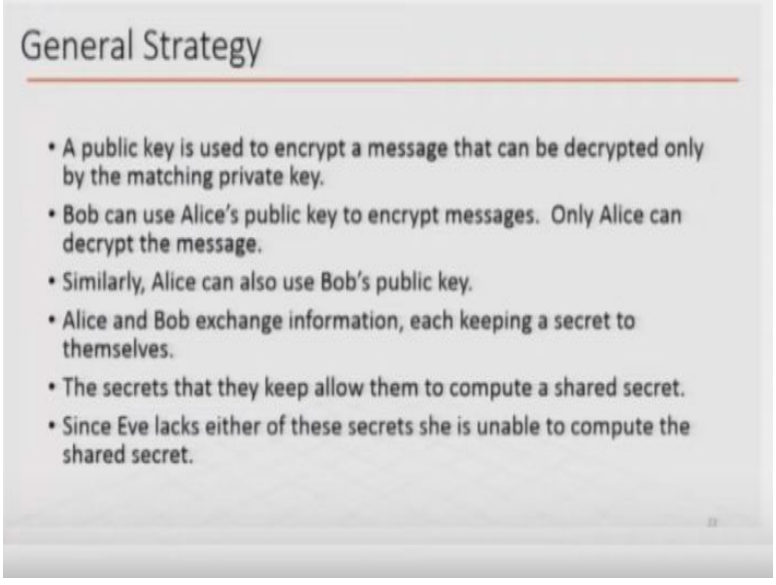
And then if you do not tell people p and q, and just give the n for people to know the p and q is hard. So reversing the process is difficult.

(Refer Slide Time: 33:23)



So public key cryptography as you know that the person who wants to send you the message, you basically send it encrypted with the public key. And then, when you decipher the message, you do it with the private key.

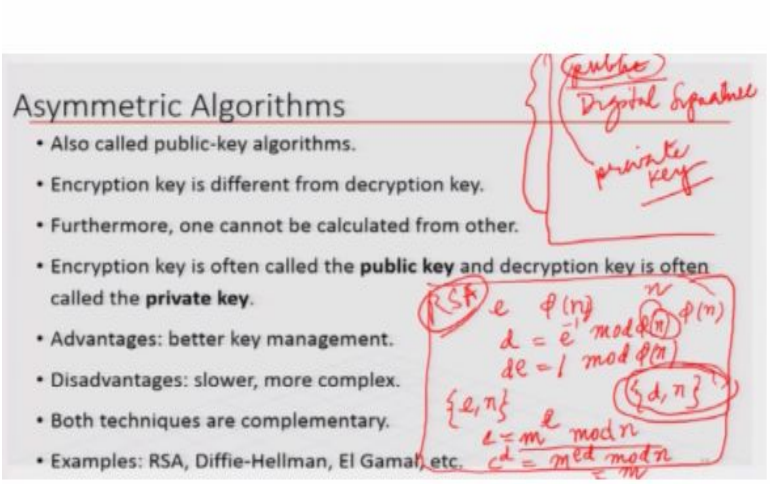
(Refer Slide Time: 33:38)



So this is what we already talked about. The public key is used to encrypt a message that can be decrypted by matching private key. So Bob can use Alice's public key to encrypt messages sent to Alice. Alice is the only one who has the private key, so she can decipher the message. And Alice can also use Bob's public key to send message back to Bob. And Alice and Bob can exchange information, each keeping a secret to themselves.

And the secrets they keep allow them to compute a shared secret. And so since Eve does not have any of the secrets, she is unable to compute the shared secret. That is the general strategy.

(Refer Slide Time: 34:23)



So this public key algorithms there are many. So for example, RSA Rivest-Shamir-Adleman algorithm is one of the most popular algorithm for public key or asymmetric key cryptography. There is also Diffie-Hellman, there is El Gamal. So there are multiple different algorithms. And encryption key is different from decryption key, one cannot be calculated from each other.

And encryption key is called the public key and decryption key is called the private key and you can do better key management. And disadvantage is that is slower and more complex. And both techniques are complimentary, which means that asymmetric is used to establish a shared secret, which is then used as secret key for symmetric algorithm. So let us quickly tell you what how the RSA works.

So in RSA what you do is, so one party decides two very large prime numbers, p and q. Prime numbers, you know that, if I guess an integer to check whether it is a prime or not, is a polynomial time, and due to an algorithm that was invented in IIT Kanpur by Professor Manindra Agarwal, Nitin Saxena and Neeraj Katyal. Now you take two large primes p and q, and then you multiply them and you get a number n.

Then you compute another number, p - 1 times q - 1 and you call it phi n and then you forget this numbers. So you do not tell anybody the numbers, throw away these numbers and you have the n and phi n, right? Then you choose a number e, which is mutually prime with phi n. Mutually prime means if you take the GCD of e and phi n, that should become 1, so they have no common factor.

Then you compute d and the d is a multiplicative inverse of e. In other words, d times e is equal to 1 modulo phi n. If you do not understand modular arithmetic, that is fine. We are just giving you this information, it is not necessary for this course to actually know how the algorithm works. But just to give you an idea, so then the public key is e, n and it is published. And the private key is d, n, which is kept secret.

And then when you want to get a message through, what you do is

$$\mathbf{m^e \bmod n}$$

Now you say how do I do this because message m is basically any anything in inside the computer, is a string of bits. So you can actually think of the message as a large number or you can chop the message into multiple chunks, which are each are smaller than n.

Because otherwise you would not be able to do modular arithmetic, and then you raise each of them by e. So that becomes your ciphertext. So when the message goes on the other side, the one to the one who possesses the private key that is this, then he does $\mathbf{c^d}$, which becomes

$$\mathbf{m^{ed} \bmod n}$$

ed being

$$\mathbf{1 \bmod phi(n)},$$

this becomes basically **m**. So I keep **d** and **n**.

So nobody else knows d, and it is difficult to know d unless you know phi n. But how do you know phi(n) because

$$\mathbf{phi(n) = (p - 1) * (q - 1)}.$$

To know phi(n) you have to factor n first into p and q, then you can get p - 1 and q - 1. So that is the whole idea of RSA asymmetric cryptography. So this is how you establish a shared secret, and then you go on doing things.

But what we will see is that there is another use of this, which is suppose I want to prove authenticity. See earlier I said that in symmetric key cryptography if I have shared the key with you, and if I get a message that is encrypted by that key, I know that it is authentic because you are the only one who knows the key.

In public key that is not possible if somebody sends me a message encrypted with my public key, how do I know it is it is not the one who is claiming it to be and somebody else because public key is known to everybody. So anybody actually can send me encrypted with my public key, right? So therefore, I do not have authenticity, like that.

So what I do to prove my authenticity is I use my private key to encrypt something and send you. And you know my public key because everybody knows my public key. So you use the public key on that message, and you decipher the message. Now you know it is me because I only have the private key. If it was not encrypted with my private key, then my public key would not have unlocked it.

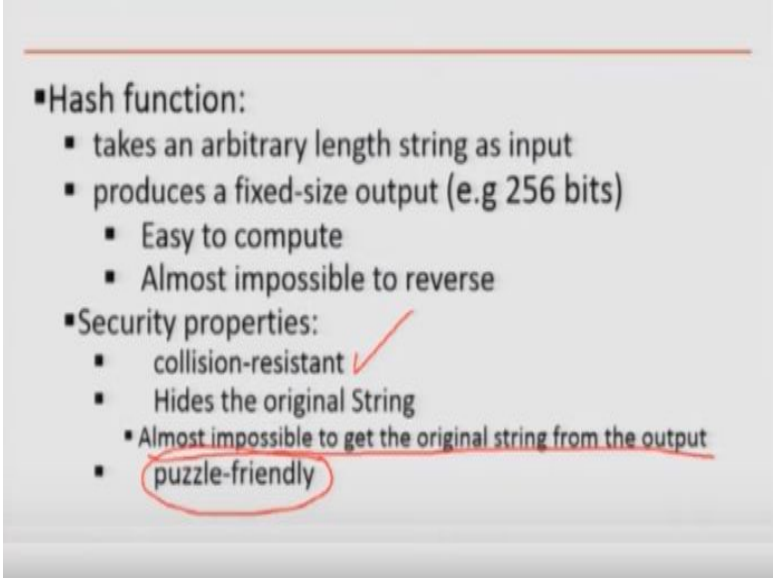
So this is how an digital signatures work. And we will talk a little more about that later. So we will see a number of things that are very useful in blockchain for from

public key cryptography. One is the public key. Public key for each person is very important. And the digital signature is very important. And because public key is there so private key is also very important because digital signature is done with private key.

Also anything that is encrypted with public key will have to be deciphered with private key. So these concepts, as we will see will come in blockchain all the time. And they basically constitute a key backbone in the technology of blockchain. So examples as I said are RSA, Diffie-Hellman, El Gamal, etc. So now we go into the idea of cryptographic hash functions.

So far we were only talking about encryption and decryption and encryption and decryption are used for confidentiality, for authenticity, etc. For integrity, some of the times we have to use a different technique. And we will see how hash functions are used.

(Refer Slide Time: 41:42)



So hash function is a function that takes any length string and creates a fixed length string. So if you give me a million bit string, I can apply a hash function and let us say the hash function is a 256 bit hash function, then it will give me a 256 bit output. So if you give me even larger, let us say 10 million length string, it will still produce me a

256 bit output. So and the property of hash function should be such that it should not take too long to compute them, because we have to use it time and again.

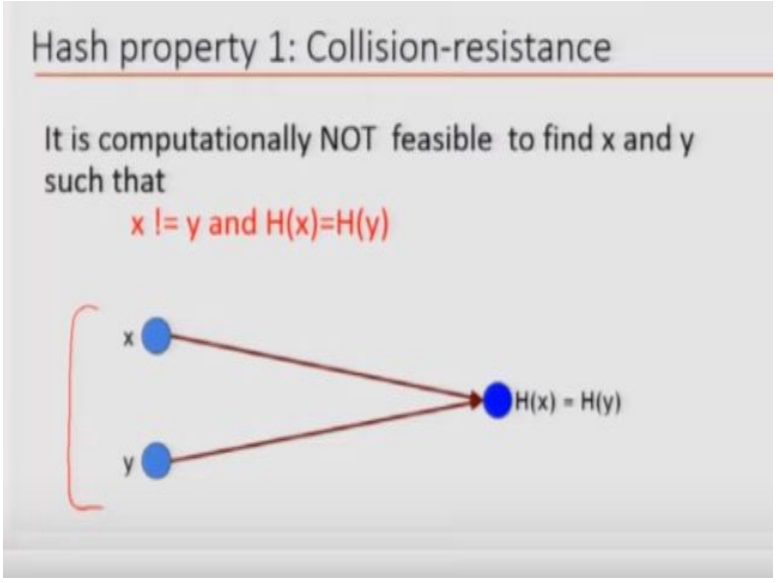
So it has to be easy to compute, but you cannot reverse it, which means that if you if I give you the output, you should not be able to say which you know string was put into the hash function to produce this output. So it is almost impossible to reverse. Now it has to have some security properties. So any function that makes it impossible to reverse is not necessarily a good hash function. So it has to have certain properties.

So one important property is collision resistance. So it must be and I will explain to you what collision resistance means in a few slides down. But collision resistance is one of the most important property of hash functions. And we will see why. And certainly it hides the original string. So if you are going to say I want to prove I know this string. But I do not want to tell you the string.

Then what I can do is I can send you the hash. And then the other party should have a way to check that this is the hash of that string. So even if the other party does not know the string itself, but he might actually somehow have the hash and therefore he can match them. So it hides the original string. And as I said that the hash property, the property of a hash function should be that it should be almost impossible to get the original string.

And then the third property that is very important for our purpose in blockchain especially for cryptocurrency type of blockchains is that it has to be something called puzzle friendly. And we will see what that means very soon.

(Refer Slide Time: 44:09)



So collision resistance, what does it mean? So it means that if you have x and y, and their hash is the same, then we say that x and y collides, means their outputs are same hash. So now the question is, if I get a hash, can I find two distinct x and y, whose hash is that particular string. So I take a hash value. And I ask you that give me two strings, which are different, but they are giving me the same hash output.

And that saying that finding a collision and a good hash function should have the property that it is not feasible to find such x and y, who collide through this hash function. So now you say why is this important? It is important for the following reason. So when you download a open source software, for example, the originators of the software, so when you are downloading over the internet, it is possible that the bits that you are downloading might get corrupted.

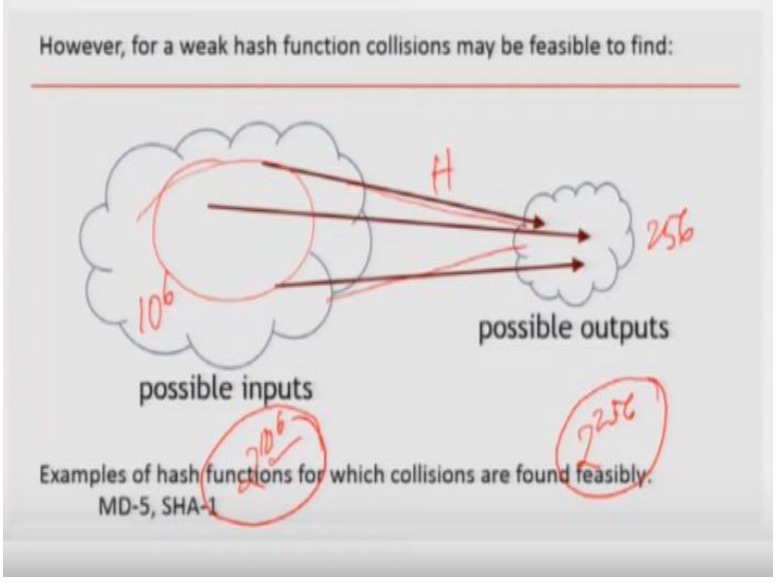
So therefore, all of these things come with a hash. So the entire file is hashed and it is sent with the file to you. And your computer then checks by applying the hash function on the entire binary and check the hash matched, right. If the hash match, then it knows that the file was never modified or corrupted on the way to download to on your machine. So therefore, the hash here works as some kind of a signature of the file.

So if hash is not corrupt and if the binary is not corrupt, then binary's hash should match the hash that came with the file. Now suppose a cyber attacker wants you to download a malicious file and somehow makes you believe that this is the real file. So then you download that file. And then if you know the hash of the original file, then your computer will say that okay, so I the file that I have downloaded, I hashed it.

And its hash does not match because I have got the hash of the original file elsewhere. And it is it does not match. So this is how we use hash to detect the integrity of a file. Now suppose I am an attacker and I know how to find collision in the hash function that is being used. Then what I can do is I can suitably modify the file, put my malware content in the file and suitably modify so that the hash of the file, even though the file has changed, the hash of the file remains same as the original file, which means I have found a collision.

And if I find a collision, then hash is no longer a good method to check the integrity of the file. So therefore, the hash function should be such that it should not be easy to find collision and this is called collision resistance. Now the collision is not an impossibility. We said it is not feasible.

(Refer Slide Time: 47:38)



And the difference between feasibility and impossibility is that feasibility means that it is not easy to compute in a fast. Impossibility would have meant that it does not

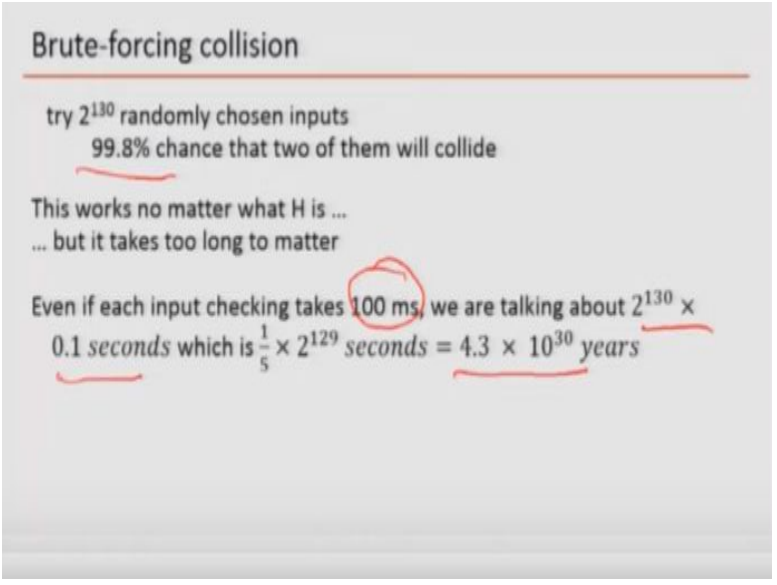
even exist, such collisions does not exist. And that cannot be the case. Because, as you can understand, I said that I have arbitrary length strings here, which are then mapped to a fixed length string. Let us say this is $\mathbf{10^6}$ length strings, and these are 256 bit strings.

Then I have all the possibilities here is $\mathbf{(2^{10})^6}$ and here I have $\mathbf{2^{256}}$, which is much smaller than $\mathbf{10^6}$. So this number is much bigger than this number. So if the hash function is mapping all the strings here two strings here then obviously there will be collisions. But the point is, so it is not like if the collisions do not exist. So we do not say a hash function is collision free.

We say a hash function is collision resistant, which means that people should not be able to quickly find a collision. So that is collision resistance. Example hash functions on which collisions were found are MD-5, SHA-1, SHA is the secure hash function, version 1. This have been now abundant as suitable hash functions because people found that collisions for this functions can be found.

So until it was found that they have collisions easily in a feasible, then they were used quite a bit. But as we know now that they have collisions, and it is not difficult to compute the collision, so therefore, they are not being used that commonly.

(Refer Slide Time: 49:21)

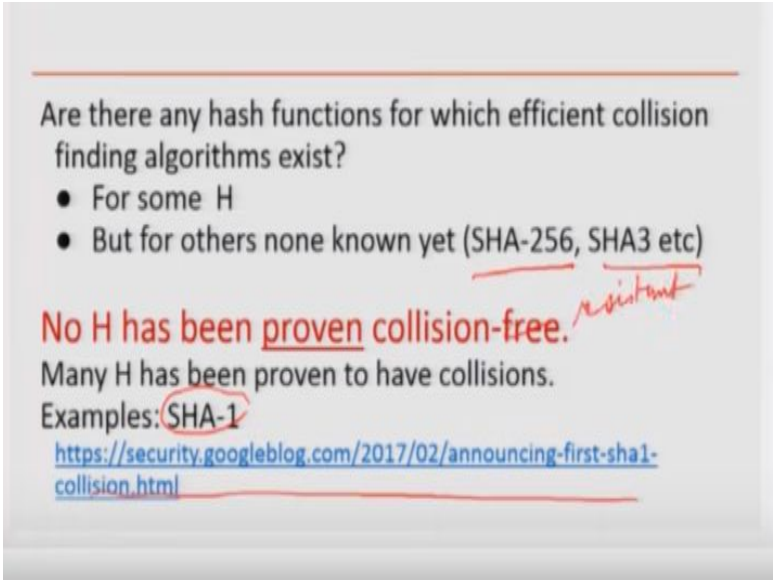


So collision, of course can be found by brute forcing the collision. So if you have a 130 bit string mapped to a let us say whatever number, let us say a 64 bit hash, then you have to try 2^{130} possible inputs and apply hash to each of them. And you will find collisions because collisions do exist. So you can compute that you know 99.8% with chance you can compute these probabilities with simple combinatorics that collisions will be found.

The point is that suppose for each input and making hash of them and so on takes let us say hundred millisecond, then we are talking about $2^{130} * 0.1$ second, which I computed approximately, will become 4 times 10 to the 30 years. That is a huge number of years it will take to try this, right. So therefore, it should be not possible to brute force the collision.

Unless you find some weakness in the hash functions construction itself, it should not be possible to find the collision by just brute force method.

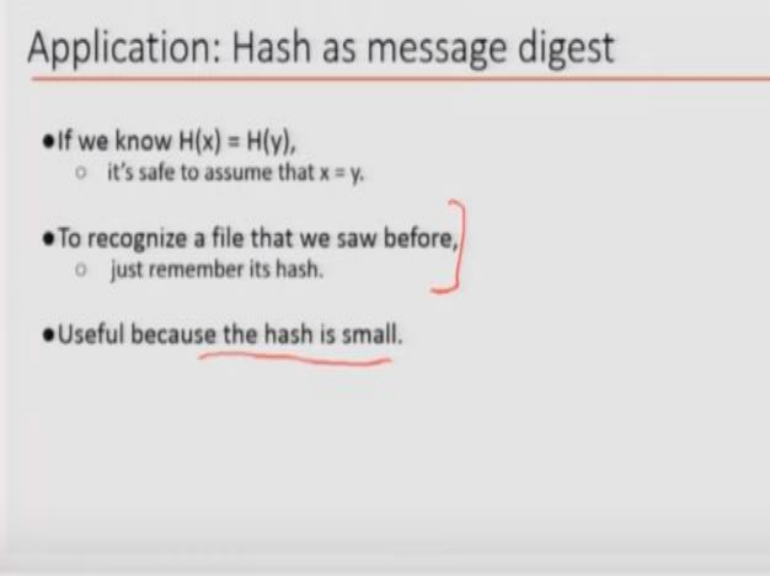
(Refer Slide Time: 50:37)



So there are hash functions for which efficient collision finding algorithms exist. For example, the SHA-1 and here is a link that you can go to see what how the researchers found that SHA-1 collision is 256 SHA-256, SHA3 etc. are safe. That does not mean they are collision free or collision resistant. But it only says that so far we have not

found a weakness in this functions that will make the collision easy to find. Okay, so now we will look at some applications of hash function.

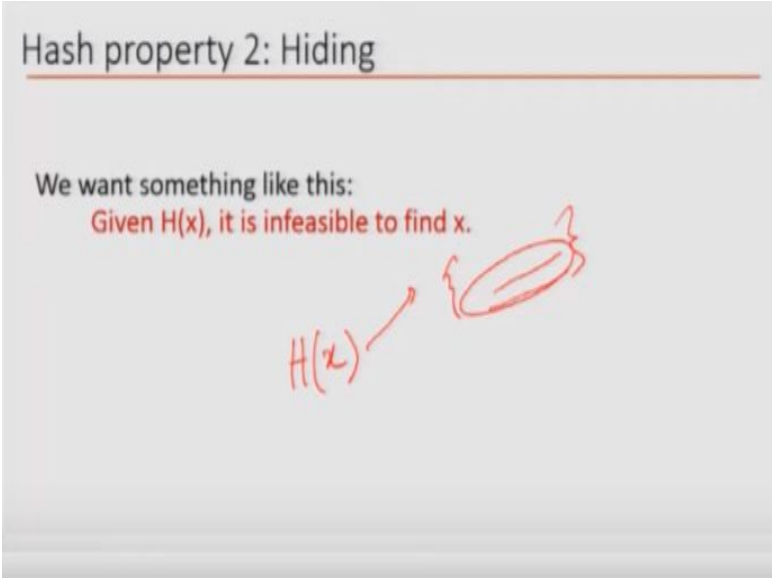
(Refer Slide Time: 51:19)



So hash can as I already alluded to, hash can be used as a digest of some very large message a file. And the reason is that since the collisions are not common, so we will assume that if hashes match, then they are they must be the same. So when I get a file downloaded, if the hash matches then I know the file is fine. And this is based on the assumption that nobody has found a collision and replaced the file with a collision file.

So it is often used for recognizing a file that we saw before. So we when we have antivirus, the antivirus tries to figure out whether this virus has been seen before. So if this virus has remained unchanged, then by computing the hash, they figure out if that virus has been already in their database. And this is very useful because hash is usually small, like 256 bits or 512 bits, so it is easy to use.

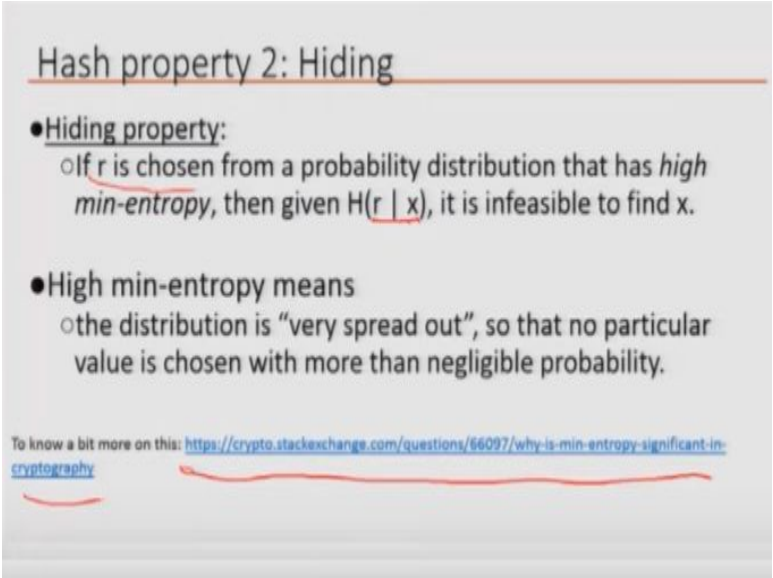
(Refer Slide Time: 52:16)



Second property is of course, hiding. That is given the hash, finding out the original message or original file is infeasible. Again, it is not impossible, but it is infeasible. If there is a hash function for which it is not infeasible, it is easy to find, then it is a bad hash function. For example, suppose you have a hash functions that maps things to only finite number of possibilities, right.

So in that case, you can create a table for every x from your domain from which you are taking inputs, and compute this possibilities and have a table. So when you see the hash, you have to just reverse lookup the table and say these are the possible inputs to which it is mapping. So unless this number is small this is not possible. If this number is very large like $\mathbf{2}^{256}$ or $\mathbf{2}^{512}$ then it is difficult and then those hash functions are workable.

(Refer Slide Time: 53:21)

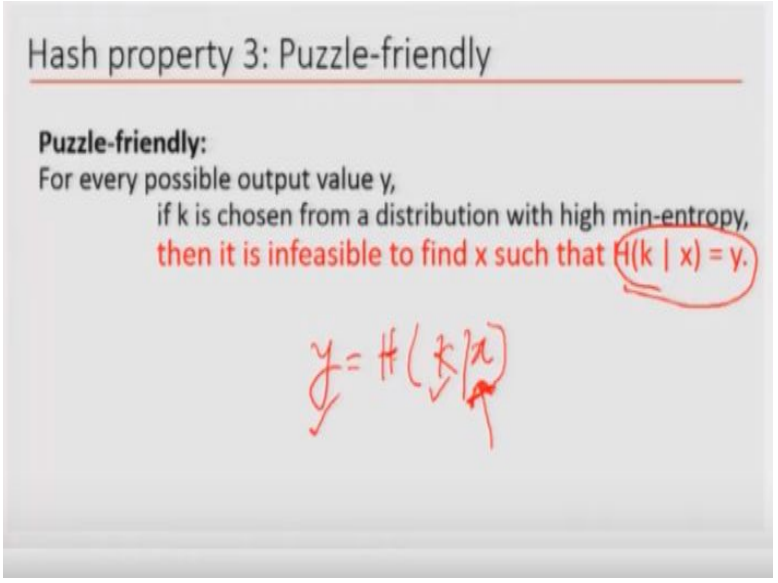


So the hiding property is more formally defined as follows. So if you have a random number chosen from a distribution with high mean entropy, then given the concatenation of r with x, **H(r | x)**, it is infeasible to find x. So what is high mean entropy means? If you want to know in a very broad terms, it means that the distribution is pretty uniform.

Because if the distribution is very skewed, then the numbers you generate may not be, certain numbers are more likely and certain numbers are less likely to be picked when you pick the random number. So the distribution should be more or less uniform or spread out so that the r cannot be guessed with high probability. And then if you take this r, pick this r and mix it with x and compute the hash.

Then finding x would be very difficult. And that is what we actually use in what we call hash puzzles. And if you want to know more about this formal property of mean entropy and why they play a significant part in cryptography, you can read various other sources. Here I have given one possible quick source.

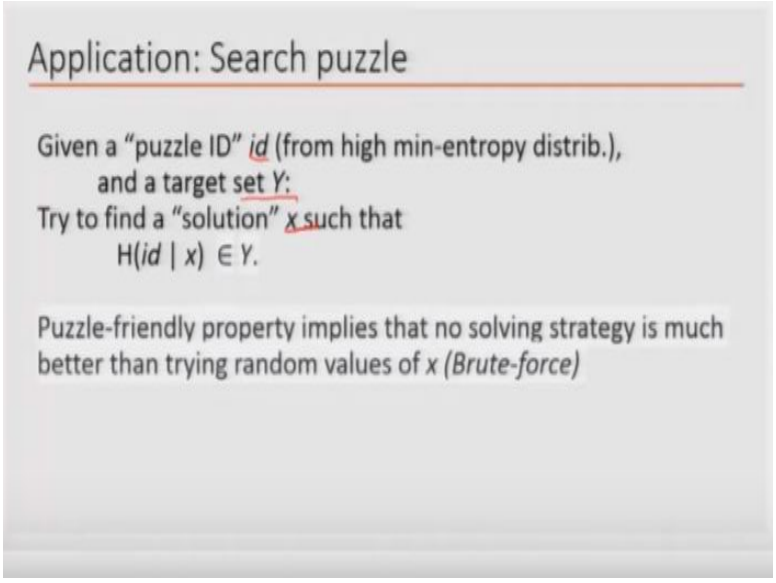
(Refer Slide Time: 54:42)



So the third property that we want to emphasize is puzzle friendliness. So what is puzzle friendliness? Which means that given an output, right, of a hash function, and given that the part of the input we know and it is a random number, then finding the rest is very difficult or infeasible. And that is the whole idea of a puzzle friendliness.

So if I know the hash, and if I know part of the the input, and then I concatenate it with some x and compute the hash, and I get y then given y and given k, finding x should be infeasible. And we will see how this is used very, very effectively in cryptocurrency mining. Basically, it is a puzzle that given y given k give me x, right. And the only way it should be brute forcing. There should be no other shortcut way by exploiting the weakness of the hash function, or weakness of the distribution of the k.

(Refer Slide Time: 55:56)

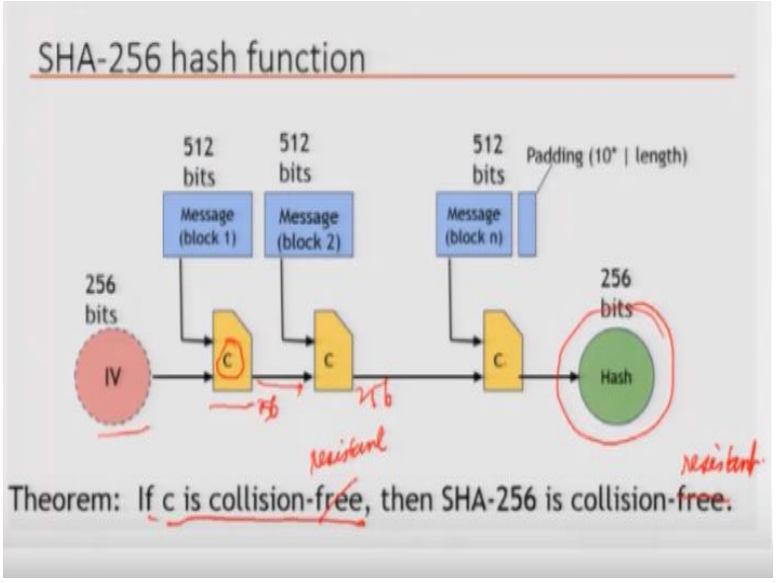


In a puzzle form we can say that given a puzzle ID and a target set y try to find a solution x such that hash of ID followed by x belongs to this y, right? Earlier we were talking about this single y given to you and k is given to you, you have to find x. Now we can also make it a little bit easier by saying that it is not a fixed y, but I will give you a set of ys.

And you give me an x, such that hash of this ID concatenated with x will give me a number in that set y. And the idea is that if the hash function is puzzle friendly, that means that there is no shortcut to find this x, except brute forcing, except trying all possible xs. I know id I know the y. And then I try 0000001 and all that stuff. And then at some point, hit a number as hash which will belong to y.

But it will take you a very long time to find out because you are going to do brute force.

(Refer Slide Time: 57:10)



So let us look at one hash function that is actually used in Bitcoin blockchain quite a bit. It is a SHA-256, secure hash function 256. This is a NIST standard. So what you do? Suppose you have a very long message and you want to create the digest. You basically use a kind of a seed on this side, and then you send, so C is a function that basically does a lot of bit shifting, slicing, and permutation and so on.

So you divide the message in chunks of 512 bits. And also at the end, your original message may not be a multiple of 512 bits, so you have to do some padding at the end to make all the chunks 512 bits. And then you apply the first 512 bits to the first function block with this seed, and then you do the output of that will also be 256 bits.

They come out as another 256 bits and you apply this in a chain until all the 512 bit chunks including the one with the padding has finished. And then what you get out is a 256 bit number and that is your hash. And there is a theorem that shows that if each of these function blocks are collision free or collision resistant, then the entire composition of this functions is also collision resistant.

So that is the idea of a function, hash function SHA-256. What you do not really need to understand SHA-256 in detail, but what you need to remember is that when you have a very large message and you want to apply SHA-256, it basically chops the

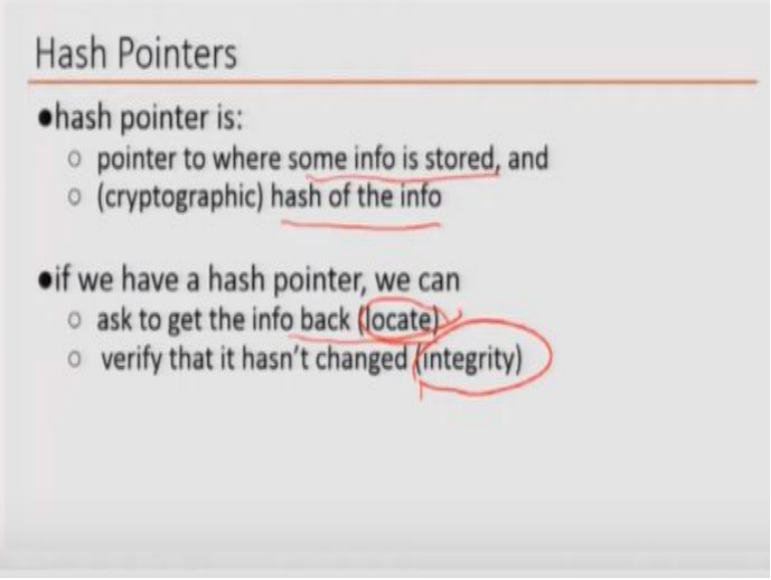
message into blocks of 512 bits, and use some padding to get the all the chunks to be 512 bits. And then it sends it through this rounds.

And each of this actually are going to be your this functions, which does bit shuffling and all. So it is easy to implement in hardware like the symmetric encryption, and therefore it is very fast to compute. It is actually can be computed very fast and with hardware support, it can be done even faster. So if you are doing in software, then you have to do a lot of bit operations and so on which is not as fast.

But if you do it in hardware, then it is really really fast. So you have lot of cryptographic hardware implemented which are which can provide you very, very fast throughput in computing SHA-256 hash. And we will see that the need for this hardware support creates this enormous hunger for electric power and for the bitcoin mining machines because they use all these things in specialized hardware.

The next thing that we want to talk about is hash pointers and data structures.

(Refer Slide Time: 1:00:28)



So what is a hash pointer? So earlier I said in the last class that a blockchain is actually a linked list. And then I said that unlike linked list that you see in C, C++ programming, where the pointer is actually a memory address for the location of the

next block of data. Here the pointer cannot be memory address because it is replicated on many machines.

And all these machines are not only the same types of machines, they are actually of various it can be a phone or it could be a very large memory machine, 64 bit machine, it could be a 32 bit machine. Therefore, you cannot say that I will point to between blocks or between nodes using memory addresses. So what you do is you have to have something called hash pointer. So what is a hash pointer?

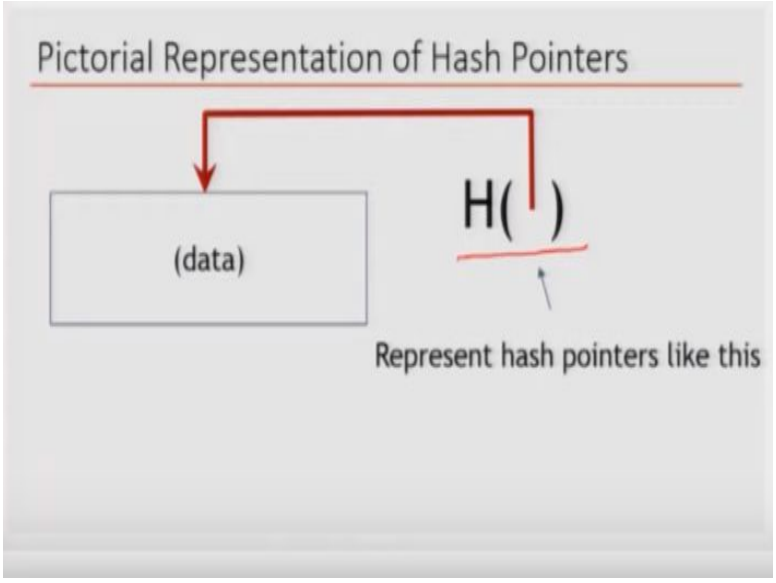
So you have a block of data and you want to locate that block of data, right. So you have to somehow point to that and keep that handle on that pointer. So what you do is you hash, compute the hash of the data and then you say that this is my handle on the data. Now assuming that for each block the hash will be different. So that hash value will tell you who are you pointing to, because in order to search that, all you have to do is that you have to hash all the blocks.

And see whose hash matches the one that you are holding, then you are pointing to that block, right? So you are locating a block by knowing its hash value. So that is why it is called a hash pointer. So it points to some info that is stored. And the way it points is by keeping the hash of the information. So having the hash, you can locate the block of data, who you are pointing to, by checking whether its hash is matching with what you are holding.

And then you compute the hash. And therefore, we as we said that hash of that data if you are holding it, and it matches that means that data has not been tampered with because if it was tampered with, then it will not match the hash. And in fact, it was if it was tampered with, then it will, you will not find the data. So if you have found the data by matching the hash that means your data was also not tampered with.

So data integrity and location can be found by using hash pointer. So although we call it a pointer, but it is not really like a pointer in the sense of C, C++.

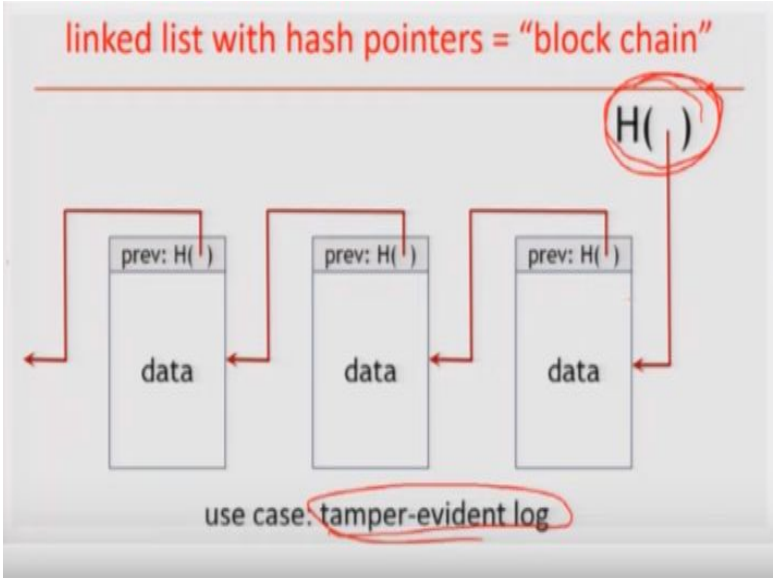
(Refer Slide Time: 1:03:12)



However, if you want to pictorially represent when we show you a blockchain, like last time also we were showing some representation of you know when you created a money with the coin creation function, then we gave it to somebody and we said that there are pointers. So we have to somehow represent pointers. So this hash pointers are usually represented by this H with this parenthesis and then this kind of an arrow.

And this is very useful pictorial representation to visualize the blockchain. So how do you build data structures with hash pointers?

(Refer Slide Time: 1:03:50)



So suppose you want to create a list of blocks and you want that like a linked list, every block should be pointed to by by the next block, right? So what you do is you

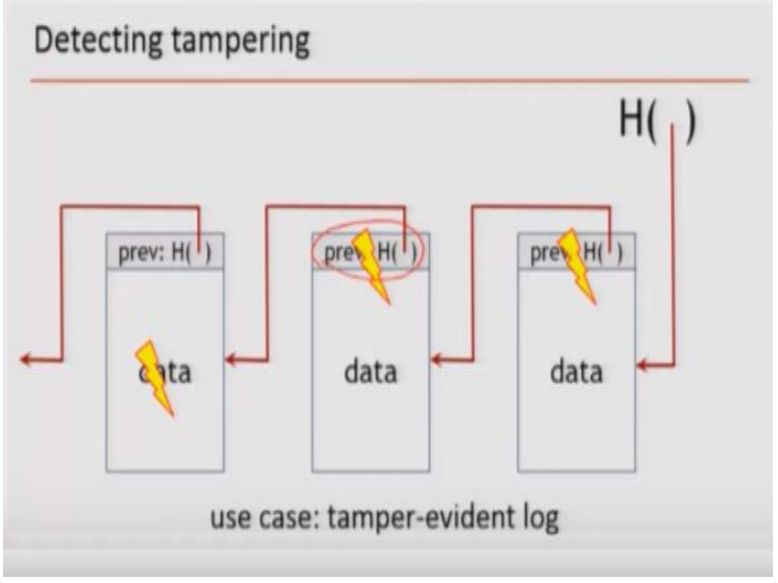
take the data and keep a part of the data space for the hash of the previous block, whoever whatever whatever is your previous block. So usually it is the temporarily previous block which means a block that was created before.

So when you create a new block, you compute the hash of the previous block and keep it inside your space. So then, this block, when it was created, it already knew what was created before it. So it keeps the hash of that here and this way, at the end, there will be a block that is called a Genesis block, which is not shown here. So before that, there will be no pointer. So that pointer will be pointing to nothing or it will be basically zero.

So so this way, if you have handle to this, then you have handle to everything because first you have to find this. Then you have to look up the hash pointer to the previous block. You use that to find this and and so on and so forth. So all you need to remember is the hash of the last created block.

And this is also tamper evident, which means that if somebody has tampered any part of the structure, it should be easily revealed by anybody who wants to check. How does that work?

(Refer Slide Time: 1:05:29)



Suppose somebody goes and changes some data in first block of data, adds or deletes or or changes some value. Now 2nd block prev hash pointer, this pointer is actually pointing to is actually the hash of first entire block(data + prev hash pointer) thing. Now if you change first block data, then the this hash the hash of this entire first block has to change. And if it is changing, then it will be easily evident to somebody who is checking the hash of first block against 2nd block prev hash pointer.

And if it finds that this is not matching, then this data has been tampered with. So therefore, the perpetrator, the guy who is attacking the system also has to do this at 2nd prev hash pointer also. As soon as he changes it here, since this is part of 2nd big block, so hash of 3rd block prev hash pointer has to be changed. So then he has to go and go and change the next block's hash also.

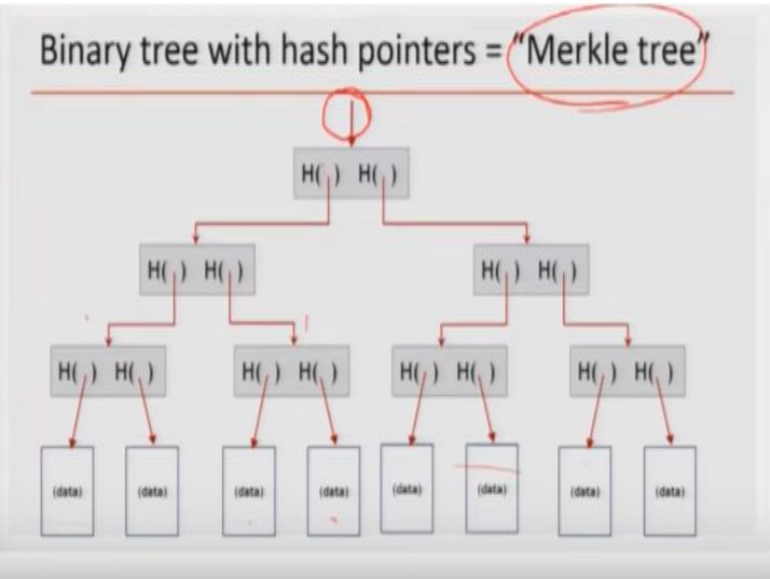
So therefore, deeper towards the Genesis block, you make change more places you have to make these changes. And as we will see later, is that you just cannot willy nilly go and change the hash or change the data. You have to actually solve a computational hash puzzle, which is very, very expensive to solve in terms of computation in order to change anything in any of these data structures right.

So therefore, if he have to change this not only he has to solve computing puzzle here once he has to computing solves another puzzle here and he has to solve another puzzle here and if this thing has lot more of these, then he has to keep solving those problems. Now to solve even one of those hash puzzles is quite difficult. So solving so many of them is computationally, you know very, very infeasible.

And that basically it gives you the tamper resistance or tamper evidence evident log property of blockchain. So blockchains are basically blocks of data which are connected to each other consecutively with hash pointers. And the hash pointer gives you two properties. It tells you who is your previous block and also it tells you that the previous lock has not been tampered with and this goes transitively to all the previous blocks, to all the way through the Genesis block.

So to change anything is going to invite too much trouble. So therefore, once the data has been stored and hashed and the hash pointers remembered, you cannot really change it.

(Refer Slide Time: 1:08:15)



Now hash pointers are also useful like any pointer when you probably did C, C++ programming you use pointers to create binary trees and you know various kinds of trees. So here also we can create various kind of data structures. So one data structure that is most common and actually there are a lot of optimization done now on this data structure. But for now let us assume that the data structure is as simple as this, which is called a Merkle tree.

If you have a block of data, you have a hash pointed to it. Now if you have another block of data, you can have a hash pointed to it. And then what you do is you combine these two hash pointers as a block and compute the hash of those and create a hash pointer to those. Now you have two more blocks of data. So you have hashes of those.

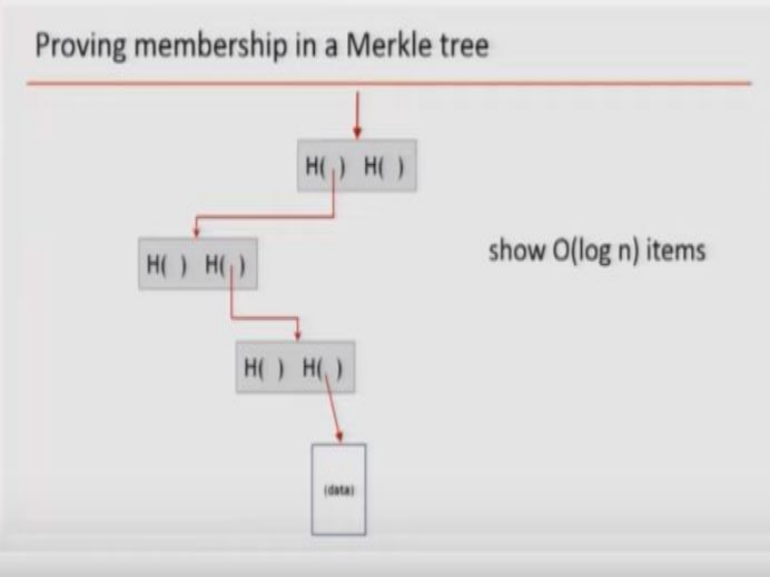
So once you have two hashes, what you can do is you can create the hash of that. So now you have another node with two hashes. Then you create a hash of that and point to that. So now if you have root hash, then you will be able to figure out child blocks and then you will find the two hashes. So then eventually can go to the data and same thing happens when you have more blocks.

So and if you want to balance it, then you do it accordingly. So this is what is called a Merkle tree. So Merkle tree, now what is the advantage of doing this instead of chaining this data in the blockchain, in the chain data structure.

The advantage is in order to locate a block, if you have a chain then you have to do a completely linear search like you have to see if you have the hash pointed to this, then you have to compute hash of the previous blocks to see which one is just before it. And then you have to compute the hash before that to know which one is before that and so on and so forth. So that is a lot of work.

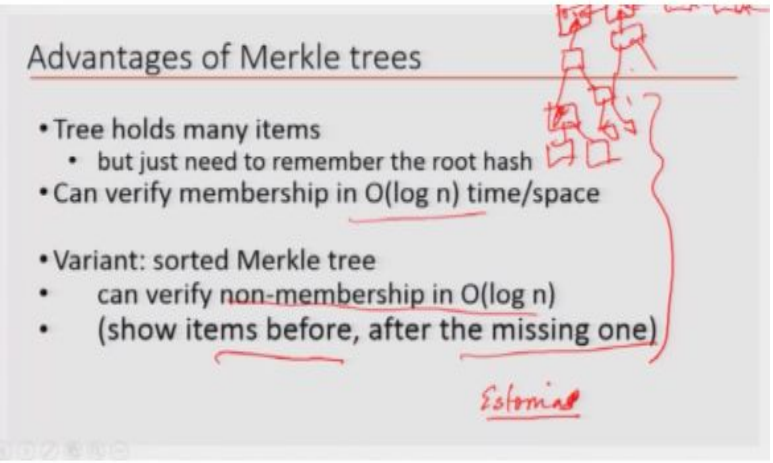
If instead of chaining them you have in the tree structure, and then you have the hash to this, then you know that this block has not been tampered with. So and then you locate this, and then you locate this, and you locate this, and then you locate this. So it is a logarithmic search for a particular block of data.

(Refer Slide Time: 1:10:39)



So to locate a block of data, you need to look at log n number of items. And whereas in a chain, you have to look at linear number of items.

(Refer Slide Time: 1:10:48)



And then the advantage of Merkle tree is that you know with n levels you can handle 2^n blocks of data. And then you only need n depth, right. So so which means logarithmic depth. So you can have verify membership. So suppose you want to know what is the membership of whether this particular data, piece of data is present. Then a logarithmic search to the trees depth from the root and then you will be finding that.

Now there is a variant called sorted Merkle tree where the members are also sorted. In that case, verifying non membership is also easy. So here verifying non membership is going to be expensive, whereas here verifying non membership is also going to be log n, because data is sorted. So if you do not find it, you can also find what is before it and what is after it.

So this kind of structure is actually used in one particular case in a very successful deployment of this hash tree technique in Estonia. So in Estonia, the government actually maintains this hash trees for all kind of government documents. And what happens is that when documents are created, they are basically put into this tree structures through the routers under which this network belongs.

And then to the next router to the next router all the way. And then every, let us say every 5 seconds, whatever is created, whatever information is created is put in a tree hroughout the country. And then the trees route is then put into a list. And so the list

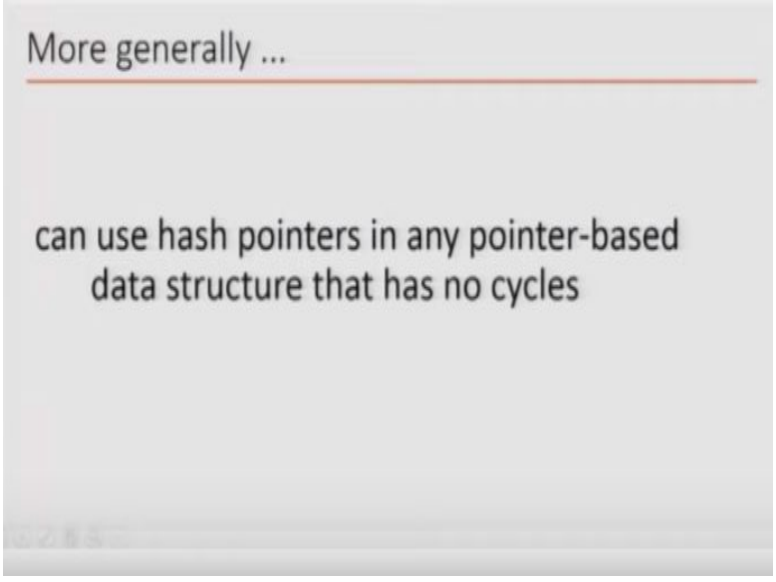
contains, let us say, every 5 seconds, whatever is created the Merkle tree root of that. Then next 5 second Merkle tree root will be that.

And these things are also hash connected. Then the government publishes the latest hash. So you have a so you have a data structure which looks like merkle tree. But these are also hash connected. So this hash connected blocks is called a calendar blockchain.

So this calendar blockchain to convince people more into the government what they do is that every weekend they publish the latest hash value. The latest hash value that has been the of the block that was created is published here in a newspaper. So if somebody changes something here some some internal government employee, then he has to change the next one next one next one and then eventually public data hash will change.

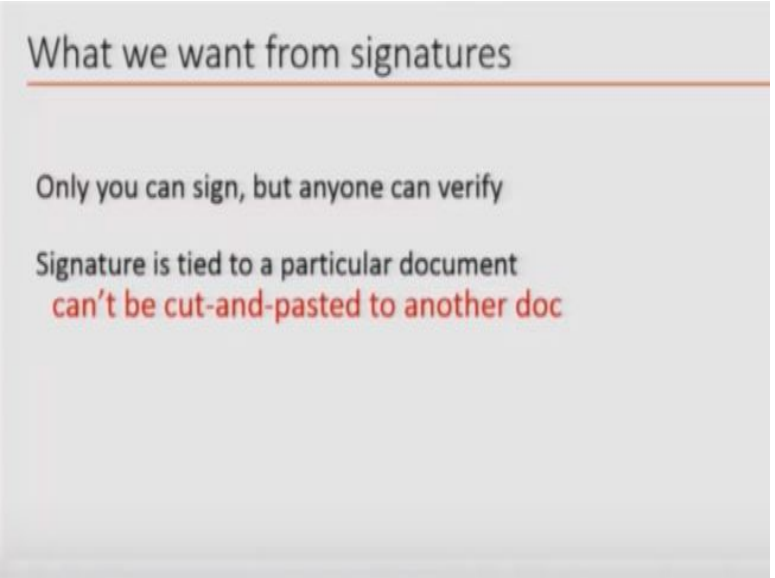
Now the public has this hash. Therefore, it will be not very easy if the public challenges that I believe that some data has been changed here. Therefore, this block got changed, therefore this block got changed, this block got changed, this block got changed. Then the government will be in trouble because public can challenge this information. So that is the idea of, you know Merkle tree.

(Refer Slide Time: 1:14:11)



So generally, any non-cyclic data structure can be created using hash pointers. So this is an important thing, so you can have more optimized data structures than even Merkle tree.

(Refer Slide Time: 1:14:28)

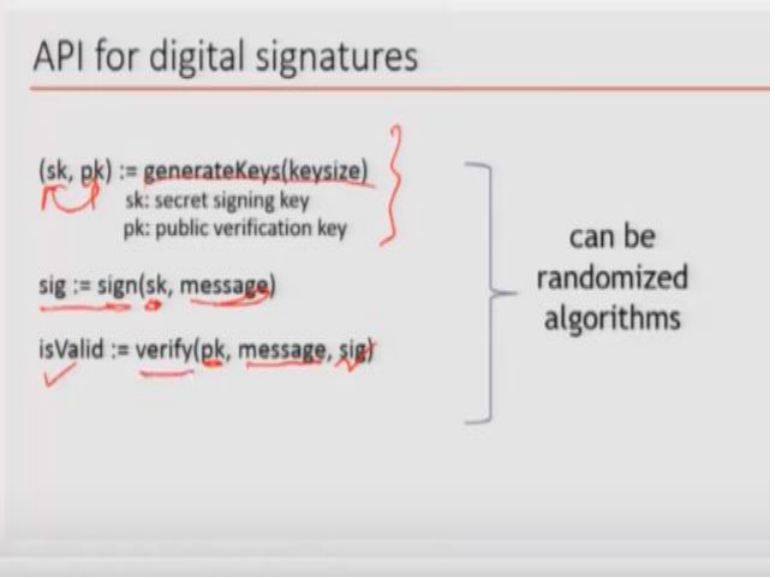


So the other thing that is very important here is the digital signatures because you always want to know in blockchain, whoever is doing transaction, the transaction before being recorded, one has to verify that it actually being done by somebody who claims that he is doing the transaction. So therefore, everybody has to have a public key and private key to be part of the blockchain ecosystem.

So I have a public key and I have a private key. Everybody knows my public key if they want to know whereas private key or secret key is kept with me. So if I sign a transaction with my private key, as I said before, digital signatures are basically some kind of an encryption done to prove that I know the private key. So anybody who can open decipher it with my public key knows that it is me who did this.

So whenever I put out a signature, then people would know that I did the transaction and if I was supposed to do the transaction, then that is a valid transaction. If I was not supposed to do the transaction, it is my signature, then the transaction will be considered invalid and will not go into the permanent blockchain. So how do the digital signatures work?

(Refer Slide Time: 1:15:42)



First, you have to have a function for generating keys. So you say that I wanted 256 bit key. So then you call the corresponding functions generate keys. It will give you a key pair, that is secret key or private key and public key or pk. So then to sign a document, you take a message or a message digest, and then you encrypt it with your private key.

To verify that it is your signature, all I have to do is I have to check whether with your public key, I can decipher the message. And then if I can do that, then I know that this signature is valid. So there should be an isValid function. So anybody who wants to do the create key pair has to use this function generate keys. Anybody who wants to sign has to use the secret key and use the sign function.

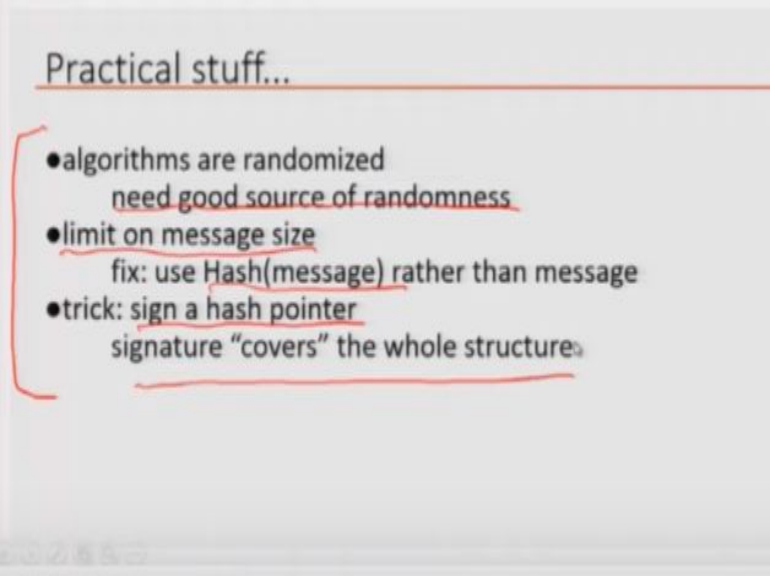
And anybody who wants to check whether it is indeed the valid signature has to use the verify function.

(Refer Slide Time: 1:16:48)

So only valid signatures, the digital signature must have the property that only the valid signatures will verify accept others will fail to verify if I am saying I am signing but I am using somebody else's secret key, then it should not verify as my signature. So you should not be able to forge somebody's signature. So unless so anybody who wants to challenge that it is my signature or not can use my public key to check this.

But he cannot produce my signature, because I keep the private key secret, unless of course my private key is stolen.

(Refer Slide Time: 1:17:24)



So this is very important part of the blockchain technology that I should be able to have a verifiable digital signature. And I have to keep my private key secret. And also

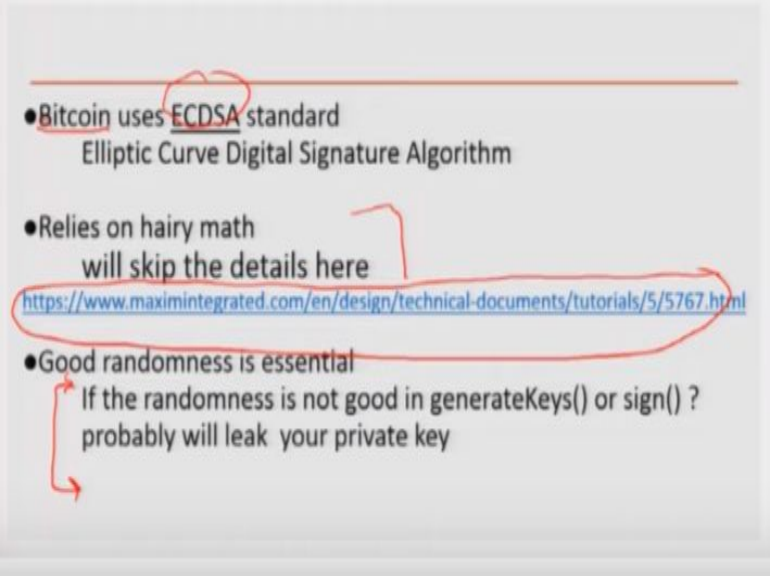
I have to be able to apply the signature on a message or a message digest. Now one important thing that I must, we must remember is that the algorithms that we use for all cryptography like generating keys and, and so on, requires a good source of randomness.

Now we know that the creation of random numbers is very difficult. And many times the function that produces random numbers turns out to be not very random. And that has been a source of problem in many cryptographic techniques. And people have been able to decipher keys by using the fact that the randomness that you thought was random, was not very random, and they could know, they could reconstruct it.

So it is very important to have a good source of randomness. Without that the whole construction of cryptography digital signature will fail. Also when you sign, you can sign a message, or you can sign the digest of the message. Because digest of the message is always smaller than the message. So that makes your signature faster.

And you can also sign a hash pointer. So not only you keep a hash pointer, you actually then sign the hash pointer and that will cover the entire structure. So hash pointed covers only the part of the structure. But if you sign the hash pointer, it covers the entire structure.

(Refer Slide Time: 1:19:14)



Now bitcoin blockchain, when bitcoin digital signature is very, very crucial. So if you use bitcoin, then it uses the ECDSA that is the Elliptic Curve Digital Signature Algorithm, which we did not talk about. It is a elliptic curve cryptography is considered more secure, and also what we call quantum resistant, which is if quantum computers come into existence tomorrow, then RSA for example, will no longer be secure.

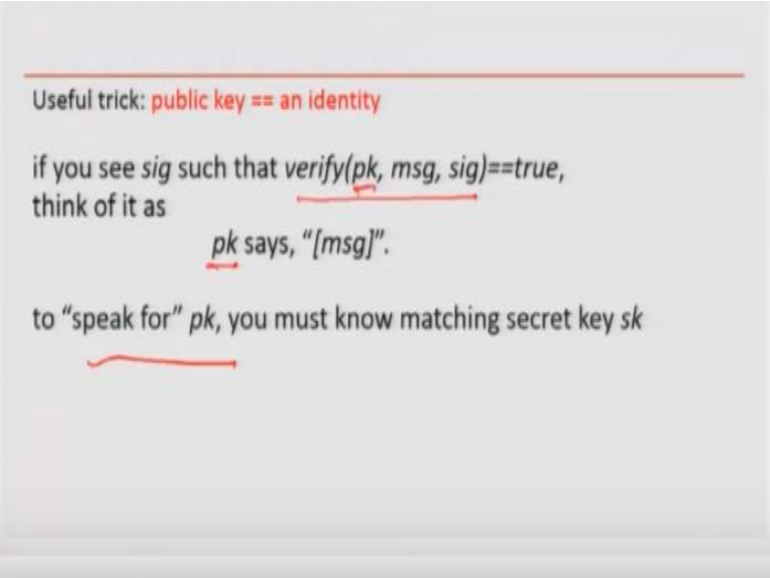
Because RSA uses the large integer factorization as its hard problem on which the whole construction is based. Now it is known that if there is Quantum Computing then factorization of large integers will be easy. Therefore, RSA by itself will not be very quantum proof or quantum safe. Whereas ECD the elliptic curve cryptography, ECC is considered to be quantum safe.

And so in that sense, Bitcoin has made a good choice, making ECDSA as your signature standard for digital signature in bitcoin blockchain. Now this ECDSA is not as simple as RSA where I was describing that it requires you to know prime numbers, it requires you to know a little bit about modular arithmetics. And it requires you to know a little bit about inverse, multiplicative inverse in a group.

And that is basically the mathematics that you require to understand how RSA works. Whereas in case of elliptic curve, curve cryptography, you need to know lot more Mathematics because you are going to do arithmetic on the points on an elliptic curve. And that you can learn if you are interested from many sources. One of the such source is here. And good randomness is a very important part.

And there is a whole area of research on pseudo randomness and random number generation and so on and so forth. But most of the times we rely on the randomness generation that comes with the system that we purchase, but if you are being very secure, then you have to worry about this source of randomness that is used to generate keys and so on.

(Refer Slide Time: 1:21:42)



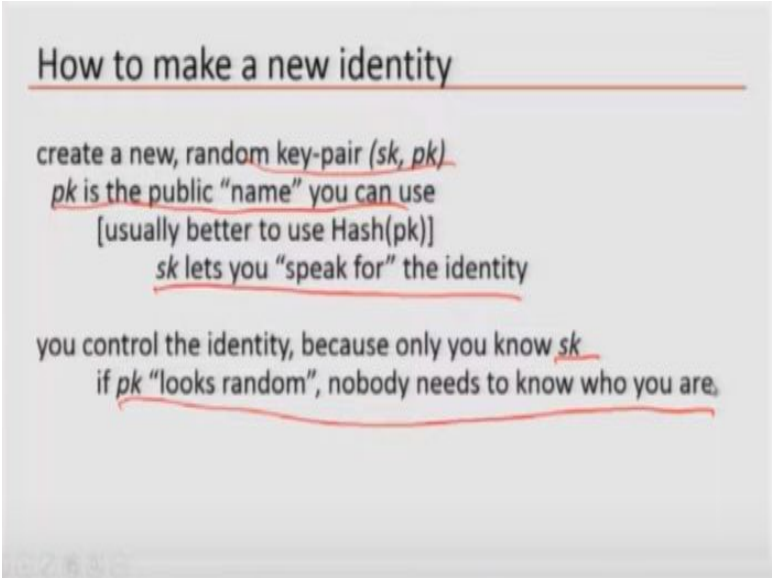
So now final thing that is, we are going to discuss today is public key can also be used as an identity and that is what is done in bitcoin ethereum and this kind of blockchains the what we call public blockchains or permission free block chains where you as a person, it does not matter who you are. As long as you have a public key, you can become a player into the blockchain ecosystem.

You can try to do mining, you can make transactions, all that stuff and nobody would know who you are. And that is basically what is done is that you have to just tell this is my public key. And so anytime you want to send me a message, you basically use this public key and knowing that will help others to check your signature.

So when you sign a transaction, saying that I am making this transaction, since you have announced your public key as your identity, so they can use that to verify that it is indeed your signature. So the public key kind of speaks for you and nobody else can speak for you or nobody else can sign for you even if your public key is known because as we discussed before, the public key is not enough to do signature.

Signature requires the corresponding private key which you should keep to yourself. So to make a new identity in the blockchain, what you do is you create a new random key pair.

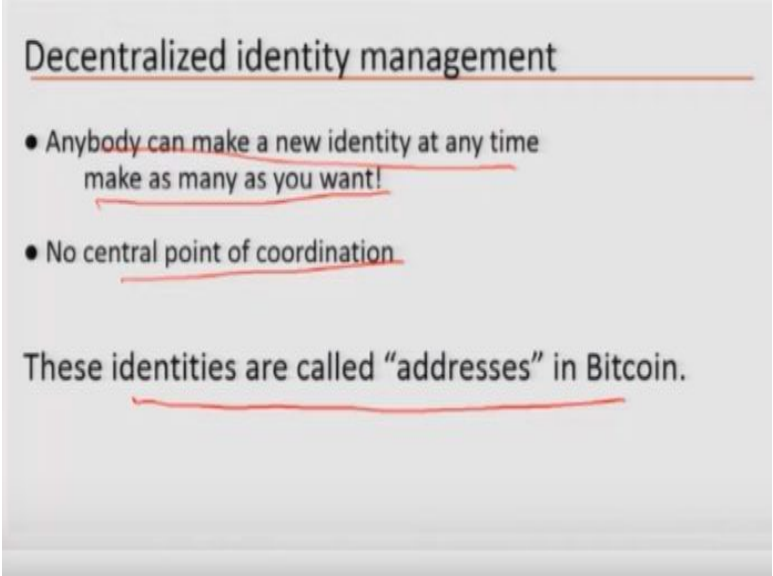
(Refer Slide Time: 1:23:09)



We know how to do that with the create key function and pk becomes your public name and sk is kept as your hidden identity or key to your identity. So if you want to prove your identity you have to use sk to sign. Now if pk looks random, nobody needs to know who you are. So therefore, one of the big problem in blockchain is that I as one person can create hundreds of different identities.

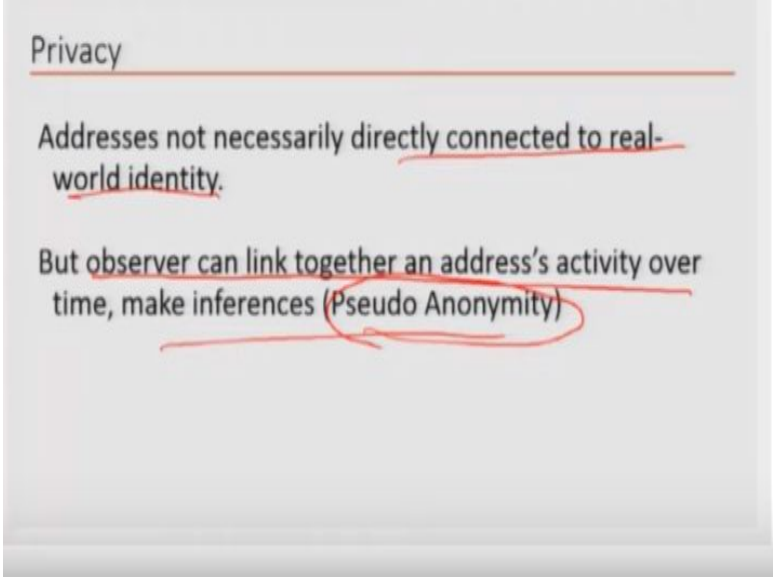
I can create as number as many as key pairs, and I can use that to create multiple identities and do various things in under the various identities. And therefore people would not be able to know that I am the person who is doing all this.

(Refer Slide Time: 1:23:58)



So that is what the problem is that anybody can make any new identity at any time and make as many as they want. And there is no central point of coordination. And that was by design, because the anonymity of users was important for the originator of the bitcoin blockchain. And these identities are also called addresses in bitcoin. Because anybody who sends money to me, he sends it to my public key and that is why it is called an address.

(Refer Slide Time: 1:24:31)



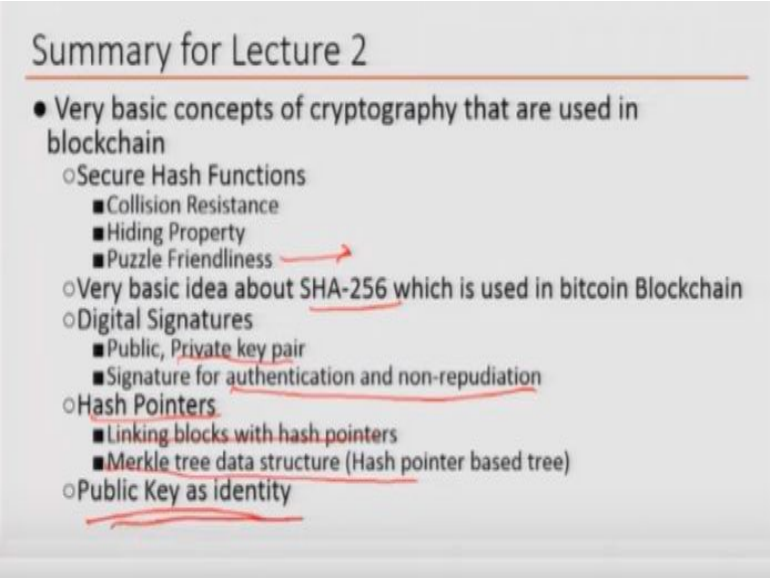
And it is not connected to my real world identity. Nobody is checking. There is no other linking or anything. So therefore, you cannot discover ever who I am. However, there may be some investigative work people can do. And if you are not clever enough, then by linking together the various addresses that you are operating, how what activity you are doing, and you are transferring money back to one particular identity and so on, they can somehow infer certain things.

And then once they know one of the identity, one of the public key's identity, real world identity, then all the other ones that are also created by you and somehow they establish the link between all this, they can make inference. So so in that sense, bitcoin blockchain is not fully anonymous. It is called pseudo anonymous.

People have shown, researchers have shown that you can actually do various kinds of data mining algorithms to see which of the addresses are probably being maintained

and operated by the same entity and things like that. But even then, discovering the real identity will require external investigative work that information cannot be found in the blockchain itself.

(Refer Slide Time: 1:25:53)

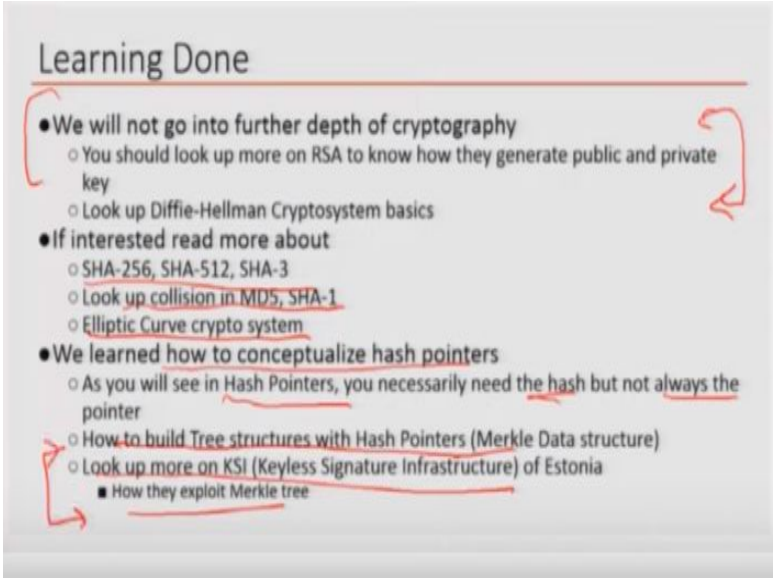


So summary of lecture 2 is that we covered very basic concepts of cryptography that are using blockchain. We talked about secure hash functions, the collision resistance, hiding property and puzzle friendliness. And this puzzle friendliness will come back when we discuss the basic operating principles of bitcoin blockchain. We talked about SHA-256 and little bit about its construct.

Digital signatures, how private keys use to do a digital signature and public keys used to verify it. And this is required for authentication and non-repudiation. We talked about hash pointers, what they are and how they are used to link blocks and Merkle tree as a data structure, which is used for efficient, locating efficient locating of data items. And finally, the public key as an identity.

So so instead of having a username and password your public key becomes your identity.

(Refer Slide Time: 1:27:02)



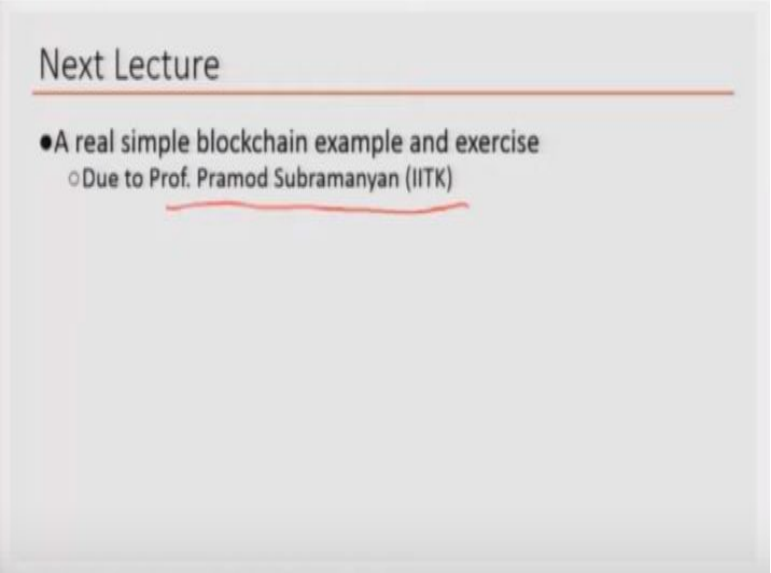
Learning done so far in the course is that we will not discuss any more cryptography. You can learn on your own more about RSA, how it works. Also look up Diffie-Hellman cryptosystem, which is another public key cryptosystem based on discrete logarithm problem another hard problem. But we will not go into the depth of cryptography because that will require an entire course and I am sure there are a lot of courses on cryptography.

If you are interested to know more about this recent hash functions, which are being used in various blockchain lookup SHA-256, 512, and SHA3, which is the latest, NIST standard. Look up the collision stories like how the collisions were found in MD5 and SHA-1. And look up a little more about elliptic curve cryptosystem. Because that is, you know ECDSA is the signature scheme that is being used in bitcoin.

So we learned how to conceptualize the hash idea of hash pointers. And as you will see the hash pointers you need the hash, but not always a pointer. In fact, the pointer is conceptual. We saw how to build tree structures with hash pointers like Merkle data structure and look up more about KSI, Keyless Signature Infrastructure of Estonia and how they use Merkle tree. That is a very instructive lesson.

You search on KSI Estonia, you will find a lot of information. I think it is a very interesting application that you should know about of hash pointer, Merkle tree etc.

(Refer Slide Time: 1:28:48)



So in the next lecture we will see how to create a simple blockchain on your own and some exercise pertaining to that. And that was this simple blockchain example was created by my colleague, Pramod Subramanyan. So we will we will see that in the next class.

Thank you.

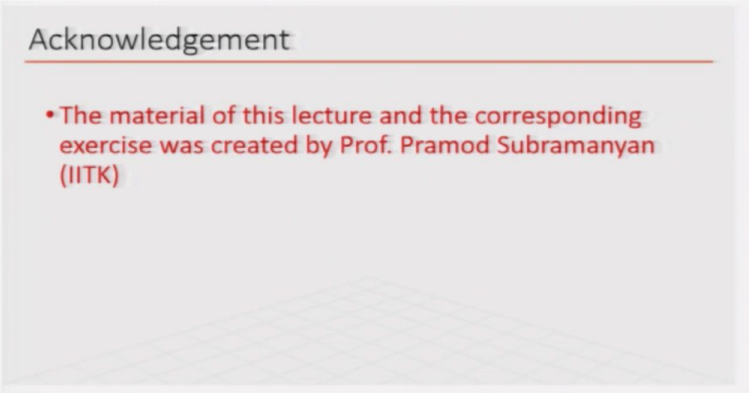
Introduction to Blockchain Technology and Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture No. 03
Blockchain Technology and Applications

Welcome to the lecture number 3 for blockchain technology and applications. So, what we will do today is announced a homework, it will not be graded but it is something that you could do yourself so that you can actually familiarize yourself with a number of concepts that are very important in blockchain. So, you will be designing your own blockchain platform implementation. This is going to be a little bit you know, simple blockchain application.

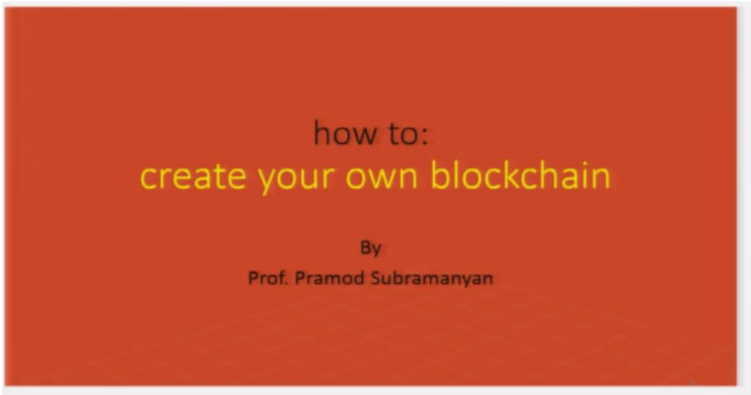
So, it is not really how the bitcoin or ethereum or other blockchains work, but it will have to come concepts that will allow you to understand the other blockchains better. Now, if you did not know C++, then this homework may not be something you could do, unless you actually learned how to program in C++ at least the minimal concepts and the data structures. But even if you even if you did not know C++, the concepts still are pretty general and you should be able to understand those.

(Refer Slide Time: 01:36)



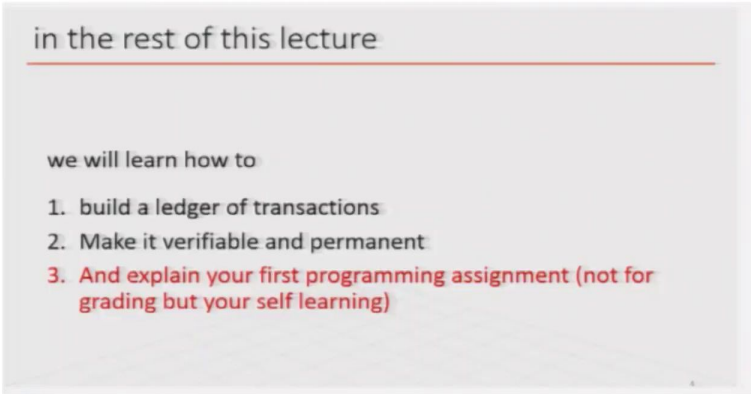
So, the material for this lecture and the exercise was created by my colleague, Professor Pramod Subramanyan from IIT Kanpur. So, we should acknowledge his help in creating this exercise.

(Refer Slide Time: 01:48)



So, the question is how to create your own blockchain.

(Refer Slide Time: 01:53)

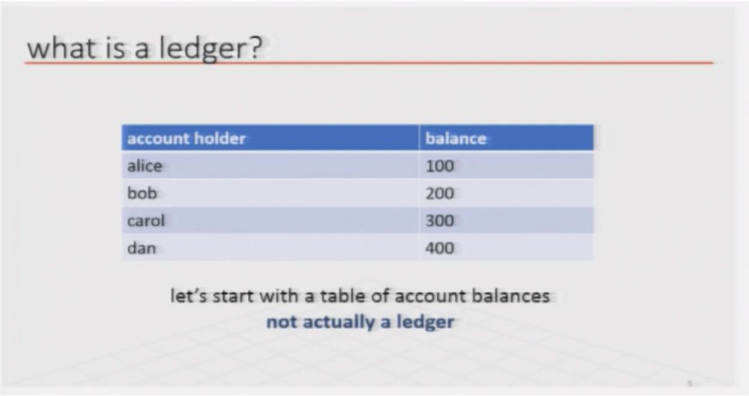


So, what we will do in this lecture is we will build a ledger of transactions and once you build a ledger of transactions, you have to make sure, that I can or anyone can verify that it is a valid, the transactions are valid, and therefore, the blockchain has everything valid. And also the how to

make this thing permanent. That is how to ensure that nobody can tamper with data once it is put in the blockchain. And then we will explain the first programming exercise is for your self-learning so we will not grade them.

If you did not know C++ you may not be able to do this however, the concepts are very important. So you should pay attention to what we are doing maybe you will have a little difficulty understanding, how we are using pointers and various kinds of data structures. But you can probably, you know, guess what we are doing, and that will be enough to understand.

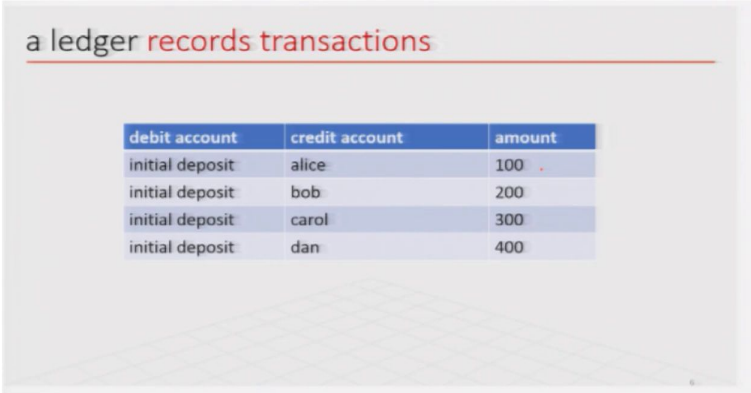
(Refer Slide Time: 03:02)



So the question is what is a ledger? So a ledger could be many things. But let us look at this. So let us say this is a bank's balance sheet. And here we have 4 account holders, and they have their balances. The question is, is this balanced table? A ledger? The answer is not really, because it is a balanced table. So it does not remember the transactions. It only remembers after a transaction, what the new balances for each of the users who are part of that transaction.

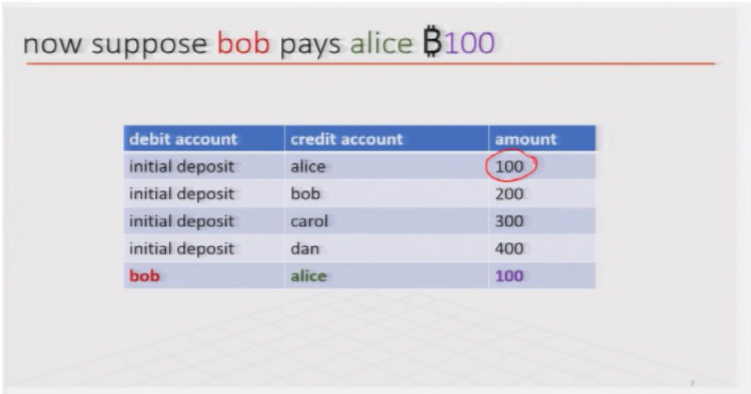
But we want to remember all transactions and we want to make sure all transactions are made by legitimate people. And we want to also make sure that the transaction actually is a valid transaction.

(Refer Slide Time: 03:54)



So now if we instead keep a record of transactions, so you should here that we have a record of transactions. So when Alice opens her account, she makes an initial deposit of amount 100, Bob makes an initial deposit of 200, Carol makes an initial deposit of 300, and Dan makes initial deposit of 400. So, initial deposit is one of the transactions. So, it has to be put in the ledger.

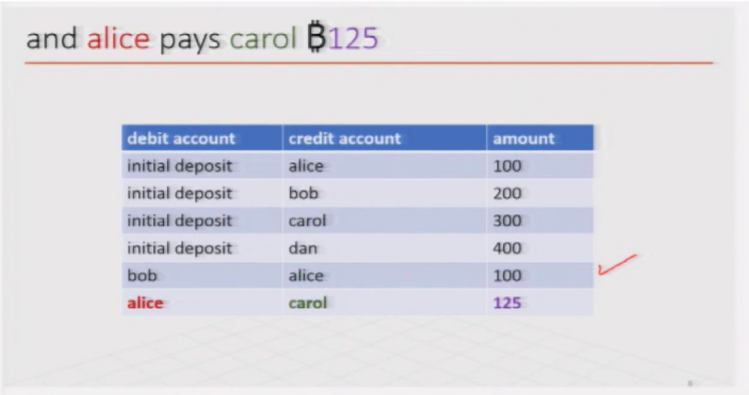
(Refer Slide Time: 04:24)



Suppose now real transactions starts happening. So let us say Bob now, for some reason, maybe Alice sold something to Bob and Bob makes a transaction maybe over the internet or by cheque, and Bob gives Alice 100 amount and when that happens, then earlier Alice used to have 200

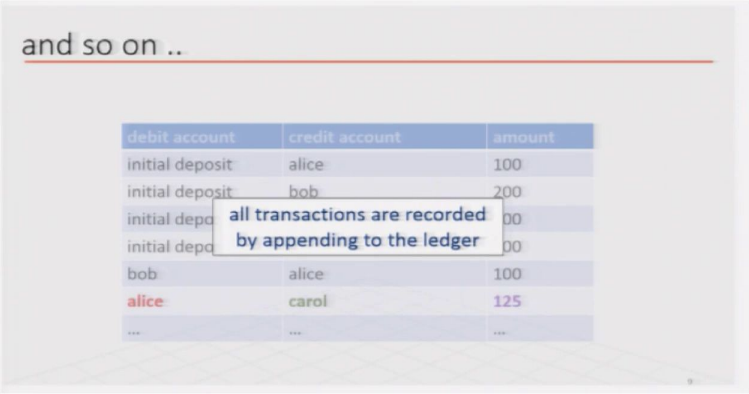
now, the Alice's account should be debited to become 200, because Bob is now giving money to Alice.

(Refer Slide Time: 04:55)



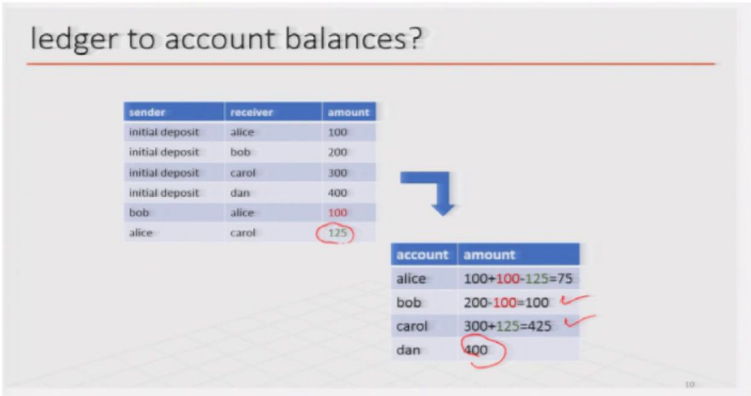
So at this point, Alice also now gives some to Carol, and in the amount of 125. So we have 2 new transactions where after the initial transactions initial deposit transactions, this is when Bob gives Alice 100 let us a bitcoins, and Alice gives Carol 125 bit transactions.

(Refer Slide Time: 05:19)



And so on so as the transactions come in, we want to add them to the ledger. And the ledger is always up and did nothing. You did not we did not normally want to, we did not go back to previous transactions and make any changes to them. So it is an append only ledger, right. And we want to ensure that this append only property is enforced and nobody can go and change the previous transactions once they have been committed.

(Refer Slide Time: 05:54)

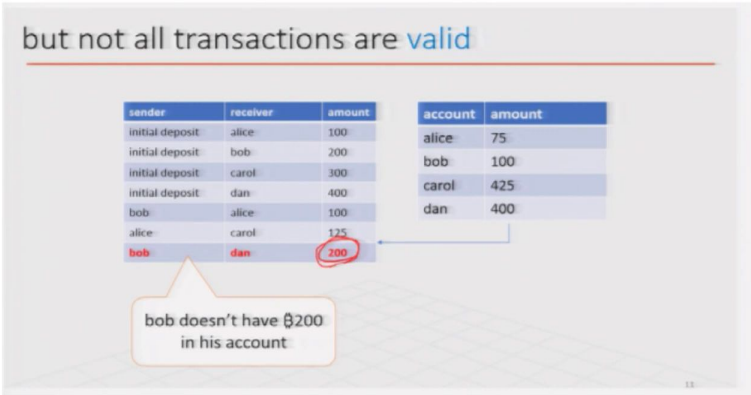


And then I can construct the account balances by going through the transaction so earlier we were we started with an account balance table. But what we are doing here is that if you keep track of all the transactions, that is enough, because at any point in time, you can actually reconstruct that table of account balances. So, I can actually see that Alice initially deposited 100, then she got from Bob another 100, then she gave 125 to Carol.

So her final balance at the end of this transaction is going to be 75. Whereas here, if you see Bob, he has given 100 to Alice. And initially he had 200. So now he has 100. Carol actually got money first by initially depositing herself 300 + 125 so she has now 425 and Dan did not get any money or give any money to anybody. So he is balanced out that 400 so, this balance table can always be reconstructed from the ledger of transactions.

So ledger of transactions is very, you know fundamental. Whereas from the balance table you cannot reconstruct the transactions. So it is better to keep the transactions in the ledger, then the account balance, because account balance is going to lose information.

(Refer Slide Time: 07:23)



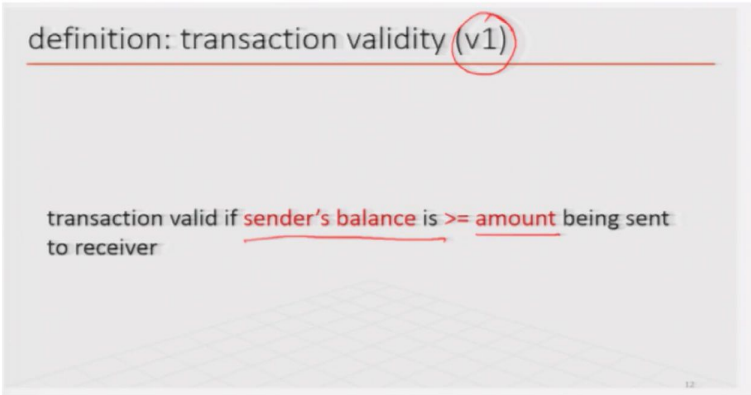
Now, when I try to do a transaction, so we spoke about that in the earlier lectures, that not all transactions are valid. For example, suppose at this point, Bob wants to give Dan 200 but Bob has an amount of 100 after all the transactions he has made earlier. So if he tries to give 200 to somebody like Dan, then that transaction will not be valid because he does not have 200 at this point. So therefore, this transaction will be rejected or should be rejected.

By the, whoever is keeping track of validity of transactions. So, in case of a bank, the bank keeps track of the validity of the transaction. But in a blockchain will see that such attempt to make invalid transactions will be checked by everybody who is part of the blockchain or whoever is a part of the blockchain and is interested in validating transaction and why would I be interested in validating other people's transaction because I want to be part of the mining system.

And so, if I validate transactions and then somehow after validating transactions, if my validation results are actually kept in the blockchain, and many others are validating transactions, and theirs is also trying to be on the blockchain and if mine goes through as part of the blockchain, then I

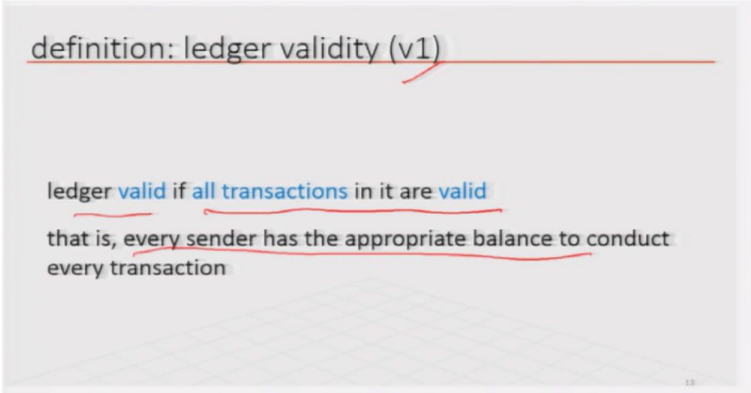
get rewards. I get there are so many bitcoins or whatever the currency is, and therefore, I have an incentive to do the validation. This is how a decentralized kind of crowdsource validation of all transactions can be met without a central authority like the bank to do this validation.

(Refer Slide Time: 09:20)



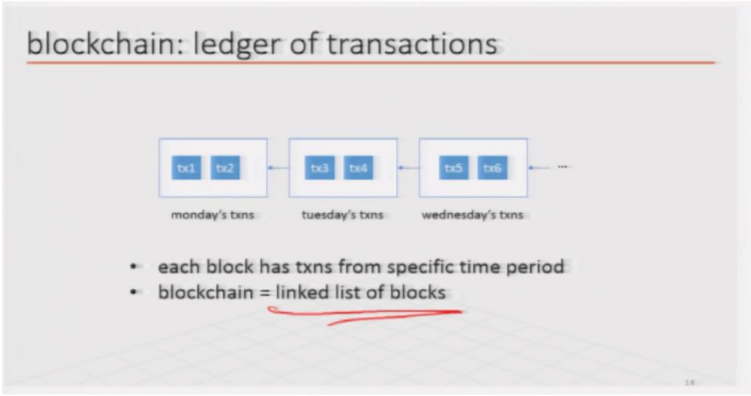
So, definition of transaction validity is now then if the sender's balance is greater than equal to the amount that the sender is trying to send, then we say that the transaction is valid, otherwise the transaction is invalid. And we call it transaction validity definition version one, because as you will see, this is not this is necessary, but this is not sufficient to declare a transaction to be valid. So we will come back and redefine or add more into this definition and get version 2 of the transaction validity.

(Refer Slide Time: 09:56)



And the ledger validity is that if all transactions that are recorded in the ledger. If they are all valid transactions, then we say that the ledger is valid. And this is also version 1, because it is actually based on the version one of transaction validity. So that is a valid ledger definition. So in other words, every sender has an appropriate balance to conduct every transaction. So that is C, and they have been recorded, and that is a valid ledger.

(Refer Slide Time: 10:27)



Now, what happens in a blockchain is that there is some further optimization. So if you start keeping every transaction, you know as a separate entity in the ledger, then and there are many,

many transactions, then the ledger grows very, fast. And remember, the ledger is has to become a linked list. So you will have too many nodes in the linked list. So what people do is that they wait for some time and bundle all the transactions that happens within that time check their validity and make a block.

So, and then these blocks are connected as nodes in a linked list. So, therefore, now how you can decide how you want to make this you know how much time window you have to wait until you put the transactions in a block. So, it could be on a daily basis it could be on a on every 1 hour it could be every 10 minutes, it could be some other criteria by which you decide when it is time to put all the transactions that you have seen into a block.

Now, interesting part here to understand is that if bank is doing it, then it has a fixed rule that I will only wait for 1 hour and every hour, whatever transactions has happened. I will put them in a block I will validate them and then I put at that block into the linked list, but if it is being done by everybody independently, so, there are let us say 10,000 people, each of which are looking at transactions and these transactions as they come at some point they have to decide.

Now it is time for creating a block with the transactions that I have seen after validating them. Now, remember this users or this node this participants in the blockchain are distributed geographically, maybe some in the US, some are in India, some are in China, some in Africa. So, therefore, their notion of time will be very because the clock synchronization is not being applied here. So, therefore, what is when I think that it is one hour the other participant from another place may not think it is 1 hour maybe a few minutes here.

And there also, the other problem is that if I make a transaction in India through the network to reach to receive that transaction happening may take more time to the US user and a transaction happening in the US, the Indian observer may actually see them later. So, what may happen is that even if you have a standard notion of 1 hour, but the transactions you have seen at this point this by observers in US and observer in India would be little different.

To the reach of the transaction information over the internet may be a little bit delayed based on the proximity of the transaction happening and the observer. Therefore, everybody when they start creating blocks, their blocks will differ. Also, be user may say, I did not like this particular transaction. So I may not include it in the block right. So therefore every user say 10,000 users are all trying to make this valid blocks of transactions.

The reason they are doing it is because they will get some reward, but they the blocks they create may differ from each other. So, then the question is, how do you decide how the system decides whose block will be the next block in the linked list. And that is where the whole mining process or the last time we talked about the hash puzzles. The hash puzzle solving, or mining process takes place. And that is how a competition starts among all the users.

And this competition is won by solving the hash puzzle first, and when the hash puzzle is solved by somebody, then the person can claim that my block is the winner and this block goes into the link list. Now, what that means is that depending on who wins, which transactions go into a block will be different. However, in this setup in in our particular programming exercise, we would not have this kind of uncertainty, because we will be doing it in from one place. Because it is just an illustrative example.

This is not how the blockchain works in terms of creating blocks of transaction and validating blocks of transaction, or adding them to the list. What we are doing here is just to get the concept of blocks and concept of linking them together, rather than completely mimicking what happens in a realistic blockchain. Therefore, for us, which transactions come into a block would be much easier to decide.

And there will be only one block created because only one entity in which case your program will be creating the blocks and adding them to the blockchain. But I just want to ensure that you understand that the difference between what we are doing in this program and what actually

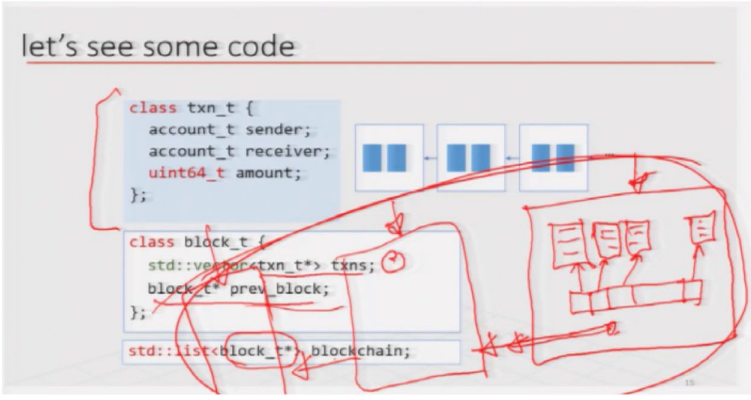
happens in the real blockchain. So, the concept of mining is absent in this exercise or in this example, only the concept of creating blocks, validating the blocks.

And validating the transactions and putting them in a list is what we are going to do here, which is sufficient to understand some of the concepts, but not all the concepts. Also, as you will remember, when I am doing it from one program and no other participants, I can create a real C++ linked list right. So, we spoke about that in the previous classes that in C++ linked list, a link between from one block to the other is actually a pointer, which is a memory address.

But we also said in the previous lectures that in case of a distributed and replicated blockchain, this memory address concept is not applicable. And there we do hash pointers here also we will do some form of hash pointer. But that is also different from how the hash pointers are used in the real blockchain. So those are the differences. So we will in this case, we will actually create a real C++ linked list or C linked list just illustrate the concept of a linked list of valid blocks.

And this linking is done through memory addresses. But with a little bit of more idea thrown into it that we will also have the hash of the previous block stored in the current block, and therefore it becomes kind of hash pointer.

(Refer Slide Time: 17:37)



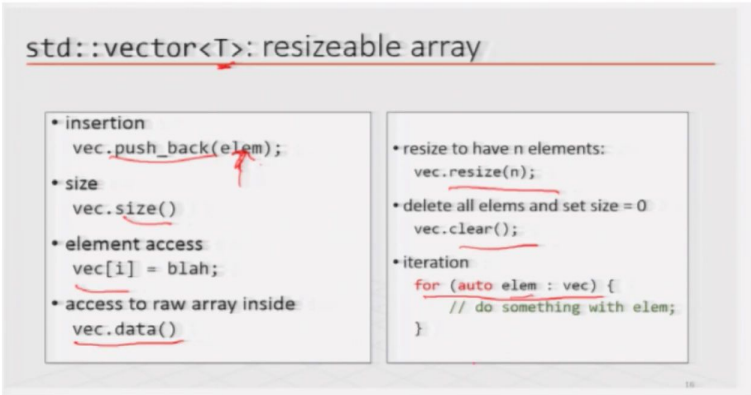
So let us see some code then in C++ we have this notion of a class is basically a structure and this class will call transaction underscore T. And this class is basically has 3 elements in the class 3 data items. One is the account information of the sender. One is the account information of the receiver and another is a 64 bit unsigned integer amount because we are not going to deal with negative amounts in transactions we are doing monetary transactions here.

So, this is a class, any instance of this class that is an object created in this prototype of this class will be kind of representing a transaction so, every time a transaction happens, we will be considering this, kind of information structure that will have sender information receiver information and transaction information. So, Bob gives Carol 100 coins. So, that is information that we want to put to that way the classes 3 elements.

Now, we have to also represent blocks. So what is a block is actually a collection of transactions and then it has to have a pointer to link the previous block. So it has 2 elements, 1 is vector of transaction. So vector is basically kind of you can think of is an array of transactions, although it has a lot more interesting properties than arrays, but it is a transaction. So it is a, it is actually a list of pointers to transactions of this type.

So, you will have a vector of transactions, which will be the all the transactions that you want to put in the block. And then you have a pointer to the previous block. So that is how the block is defined. And then the linked list then will be a list of blocks, or in this case, a list of pointers to the blocks. So we keep pointers to the blocks, and the blockchain will be a list of blocks. So that is the concept here.

(Refer Slide Time: 19:44)



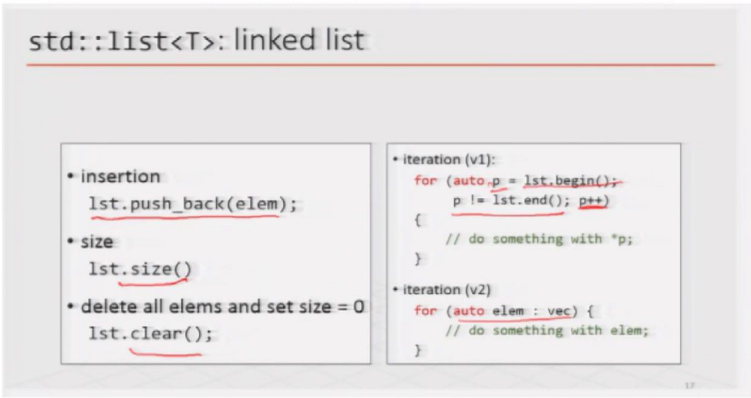
Now let us look at this data structures from the standard C++ Library. So vector is actually a template. So it can take any type T and it will create vectors of that type and if you want to add elements to this vector, so you use this function push back and whatever you are trying to push into that vector. So, this elem or element should be of the type, whatever type T you have defined. Similarly, you can check the size of the vector with the size function.

You can also access, say i th element of this vector using an indexing, like an array, so vector of i equals something would put that thing into the vectors if position. And if you want to extract the array of information inside the vector, then you can call the data function and it will give you access to that array inside the vector. Now, sometimes when you add things in the vector, so, let us say initially you define the vector of a certain size.

And then you want to add more stuff into the vector in that case because you realize that you know, initial size that you thought of is not enough, then you can call a resize function. And then it will resize it to have another n number of elements in the vector. You can also clear out a vector with a clear function or delete all the elements and set the size to 0. And then finally, if you want to iterate, if you go through the list to do some action on the elements of the vector.

For example, reading them, or changing them or playing a function on them, then this is the new way of doing it in the C++ recent standards that you basically say that elem is the element of a vector. And what this means is that you automatically iterate through the vector Zeroth element, first element, second element, third element, all that stuff inside this loop. So this for loop basically allows you to go through all the elements of the vector and do something with the element.

(Refer Slide Time: 21:57)



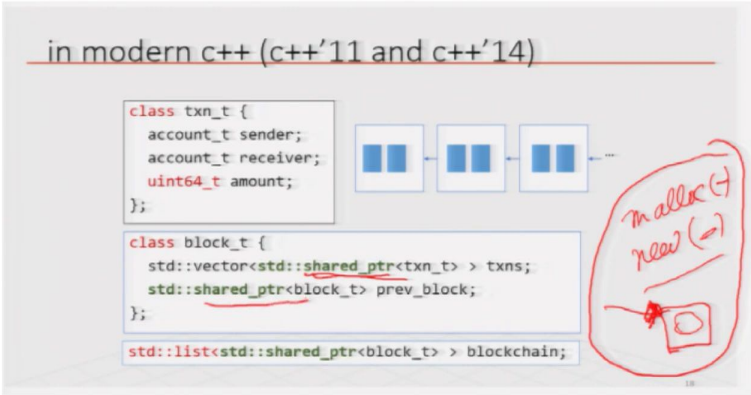
Now the other structure that you saw so vector was actually used to store transactions. And then we other thing that we saw is a linked list of blocks. So list structure was used to do that. So list has, again, certain manipulations. So in case of, you want to add an element to the list at the end, then you basically do a push back function. If you want to know the size of the list, you can call the size function, you can delete all the elements of the list by clear function, you can also iterate by various methods.

So one is that you do like you know **i = 0; i != n, i++** kind of thing. So here, it is a pointer iteration. So, you take the first pointer, and then you iterate over it, and then until you reach the last pointer in the list, so, that is one way of doing that. And the other is doing how you were doing it on the vector. You do how to do vector and do something with the elements. So for you to understand what is going on here again.

If you did not know C++, you can think of the blocks. Let us look at this so, you can think of a transaction has 3 elements the sender receiver and amount and they are put the pointer to them is put in a vector right so a vector which has first location, second location, there is another transaction, third location, there is another transaction and so on. So this is a vector of pointers to transactions. So each of these transactions has 3 elements now is that this whole thing along with the pointer is a block.

So then when the next block comes, a new block is created, this pointer is set to the previous memory rest of this block. So this is how this is done. So and then another one. And then this is a list. So this is the list of. So this, these are the pointers to the blocks. But this is a list of pointers to blocks and that is your Linked List of blocks. So, that is how we are representing the entire blockchain.

(Refer Slide Time: 24:29)



Now, another thing that the modern C++ gives you now pointed management so those of you who know C++ know that pointer management is a big headache. For example, if you define a pointer, and then you allocate some memory to be pointed to by that pointer, like using a malloc, or a function or a new in case of C++, then you are pointer is now pointing to a chunk of memory depending on what you have put here as size and everything.

Now, if you stop use of this particular chunk of memory later in your program, then you are supposed to deallocate. So, you have to deallocate this amount of memory otherwise your program will start running out of memory. Now, that deallocation is a lot of concerned, because many times people forget to do deallocate, and then we have this problem of memory leak. So the memory that is allocated to be pointed to by a pointer, it has been used for a while.

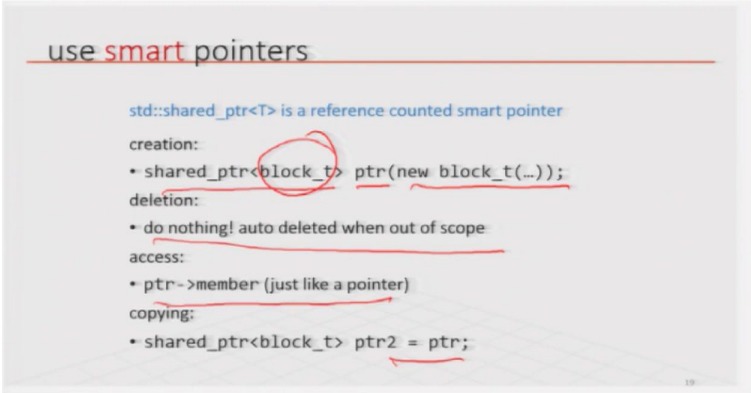
And then after a while, it is of no use, but the user, the programmer, forgot to say dealloc or the, you know, destroy or something. And therefore, the memory remains they are sitting there with nobody using them. And if it happens in a lot of such pointers, for example, in a loop, then it keeps accumulating and your memory space gets you know, gets used up and your program then will crash. So, therefore, that is a memory leak problem.

The other problem is the dangling pointer like you have a pointer for which the memory has been de allocated, but you have in your program, you start using that pointer for referencing something with that pointer and then you have a problem of null pointer exception. So, therefore, people are quite worried about using pointers. So in modern C++, there is a smart data structure called shared pointers.

Shared pointers actually are what we call reference counted pointers, which means it keeps track of whether it is under use whether the memory allocated for it is underused or not. And based on that it deallocates automatically as a programmer once you have created the pointer, you did not have to worry about deallocating the pointer, or, you know, worry about null pointer exception, because as soon as the pointer gets out of scope, or not being used, it is deallocated.

So that is why in this program, shared pointers are being used. So the only advantage here for you in the shared pointer usage is that when you are looking at you did not have to worry about deallocate.

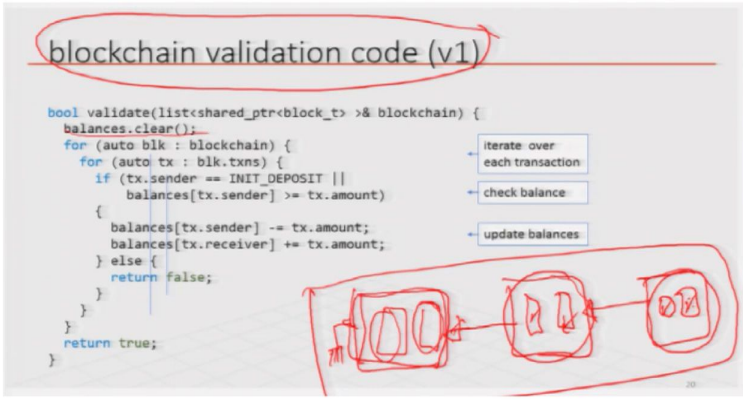
(Refer Slide Time: 27:25)



So that is the other thing. So it is a, as I said, is a reference counted pointer. So when you define a pointer, you are, going to actually define it like this. So it is a shared pointer, which points to a block, in our case, the block we defined, and then you have to define it like this for deletion. You did not have to do anything. It is auto deleted when out of scope. So that is the reason for using shared pointer.

And the usage is like a pointer right pointer arrow member name will give you access to that member of that structure which is a pointer is pointed to so, if the structure has 3 members, member 1, member 2, member 3 then you need to know the name of the pointer that is pointing to that and then you reference them as member 1, member 2, and member 3. So, that is the use of smart pointers and you can also copy a pointer to another pointer to point to the same thing and so on.

(Refer Slide Time: 28:24)



So, now, the question is, how do you validate a blockchain So, what is the validated blockchain do you have transactions etc in one block and you have next block pointing to it with transactions in them, next block pointing to it with transaction to them. And then when I say valid, I have to first check that in each of the blocks, your transactions are valid. And when I say transactions are valid, I have only defined what it means for a transaction to be valid partially as version 1.

So, therefore, this validation code is also version 1. So, first I go and validate all transactions in a block and then we declare that this block is valid. Similarly, I do this for previous block and then I declare this block is valid. Similarly, I do for previous to the previous block and I declared this block as valid. And suppose we came to very first block or the genesis block. In that case, this is not pointing to anything else. So, therefore, I declared that the entire chain is valid in the sense of the chain we have defined in this program.

So, let us see how the code works. So, first of all, in this case, we actually use a balances table, right so balances table as I argued earlier, is that I can create the balances table from the transaction ledger. Now, why do I then recreate the balances table within the program? The reason is that to validate that a transaction is valid, I have to check whether the sender's balance

is greater than the amount he or she is trying to send. So I need to quickly calculate the balance of the sender.

Now if I have to calculate the balance of the sender, every time I validate a transaction, then you will be redoing the same work again and again, I have to go through all transactions and add and add and subtract, or any transactions, any amounts that he received and subtract the ones that he has transferred to somebody else. So therefore, it is like a kind of like a caching technique. So we keep our balances stable on the site.

And therefore, first what we do is that we create a balances table. And then what you do is that we go through the transaction and check whether there is a real sender or it is an initial deposit. If it is an initial deposit, then all you need to do is that you have to add that amount to the receivers balance if it is not initial deposit, which means somebody sending money, so that the sender's balance should be greater than the transaction amount.

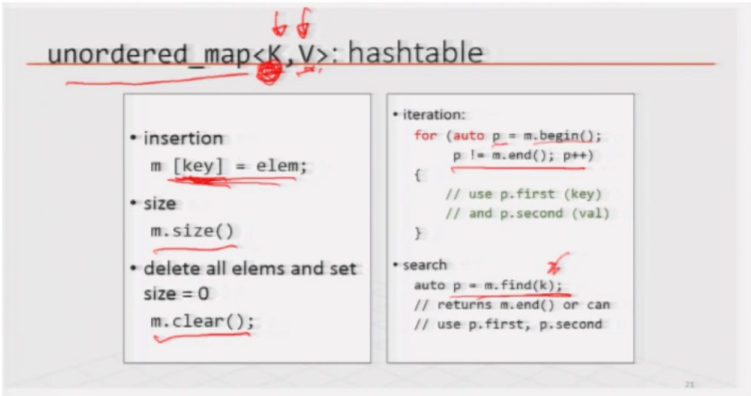
And that transaction amount should be subtracted from the sender's balances and added to the receivers balances. And if this transaction is, neither initial deposit or the sender's balance is greater than equal to transaction amount, then we return false. That means is that this code is for validating transactions. So we are basically looking at all transactions. So here you see that we are doing an auto blk blockchain in auto tx block dot transactions.

So I am picking one block, and I am going through all these transactions. Then I am picking the next block. Going through all its transaction I am picking the next block and I am going through all each transaction and for each transaction I am checking whether it is an initial deposit or the balance of the sender is greater than the amount then I do the appropriate changes in the balances and then I return false.

So, in this I have to one side effect which is the balances table at the at this current condition of the blockchain the blockchain is a growing structure, but at any point when the blockchain is at

this stage, maybe later on there will be more here, but right now, I have up to this. So, at this point, if I validate or not only I validate all the transactions, but also I have constructed the balances table. So, in the beginning of this validate call, I cleared out the balances if there was any other previous information and then we start reconstructing the balances field. So, this is how this is done.

(Refer Slide Time: 32:49)



Now, how are we keeping the balances table? So, what we have done is that we create what is called a hash table. So, and what is a hash table? Hash table is a table, which is indexed by hash. So let us say it is a table with 2 columns. In one column, you have some element. In the second column, you have the other element. And then to access the second element, like, let us say, Bob, this, Alice this so in order to know what is there for Alice.

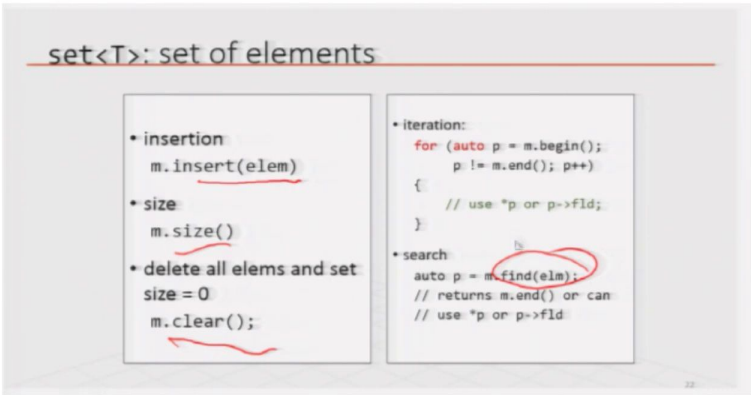
I have to index directly to Alice's name. And then I should be able to read what Alice has so in this case, if it is a balance table, it will be Alice and her balance, it will be Bob and his balance. So this is how the hash table is formed. So in this case, called an unordered map, and it has 2 different types. So K is the type of the first element or first column and V is the type of the second column. So if the first column is a string, and second column is an integer.

Like Bob and his amount, then we will have integer and will have the string and the integer. If this is a more complex structure, then let us say an address, then it will be in that structure, and it will be amount. That is how we are going to define. Now the way we map is accessed is by indexing the map with this elements. So you can think of this kind of an array, where index is used to access the element here, instead of array index, we can have our type K thing to index the array. So that is the idea.

So here also again, to insert something, you have to just say what key and what the element that you are inserting the size can be obtained by this call. You can clear the map on the table to make it reset. And you can iterate through it like if you want to go from the first elements of the table, the first row of the table to the all the way to the end. You have to do something like this like a pointer after that begins here, these are called iterators.

Actually, they are not really not necessarily pointers, and then you do this iteration or you can also search by calling a find function. So search will allow you to say, what is there for Alice, so usually find Alice as a string, and then it should tell you what the amount of Alice has.

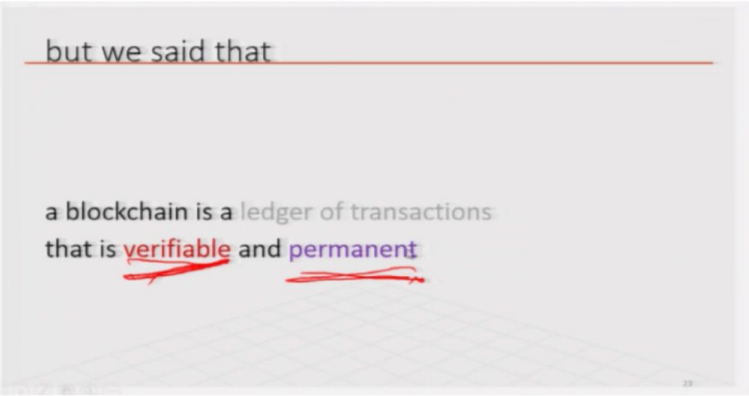
(Refer Slide Time: 35:22)



And also we have a set data structure or a template C++ template. So set as you know, is a set of elements. So, if you want to represent it in a data structure, you have to have a way to insert an

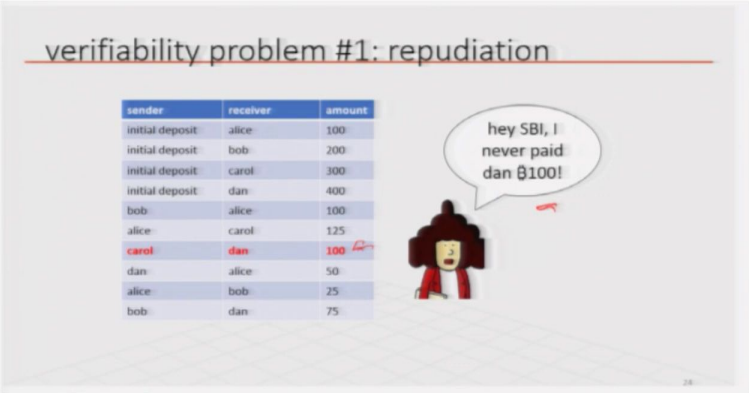
element you have to know the size that is how many elements there are in the set, you can reset the set so it does not have anything anymore. And you can iterate over this set to go from the one element to the next to the next and so on. And you can also search in the set. So, that C set data structure.

(Refer Slide Time: 35:54)



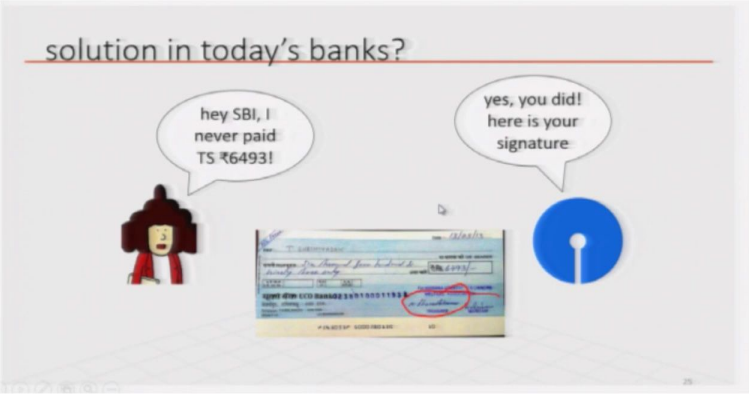
Now, blockchain is a ledger of transactions. That is verifiable and permanent. So right now, we only focused on the verifiable part that the transactions have some validity. But permanence is something that we have to now worry about.

(Refer Slide Time: 36:14)



So one issue is that we talked about repudiation a lot in the previous lectures. So let us say in the transaction found that Carol has given Dan 100 coins. Now, Carol comes back and says, I never paid Dan that money. So but why is it in your transaction, it is affecting my balance in the balance table. And so I repudiate I did not agree that I have done this transaction.

(Refer Slide Time: 36:42)

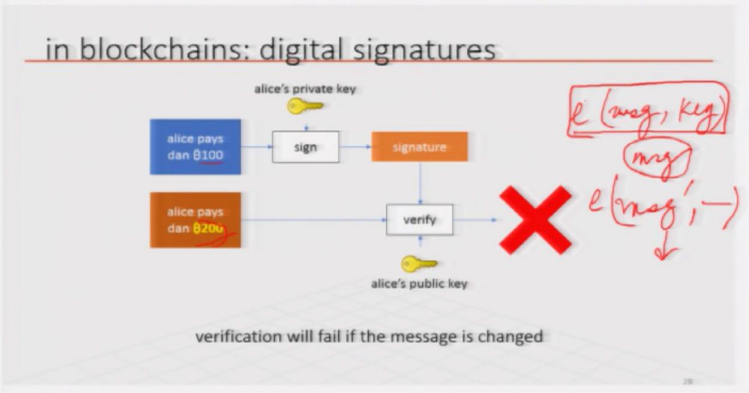


What do you do in the today's bank case? That bank comes back to you and says that look, here is the check you signed? This is your signature, we verified your signature and therefore you cannot repudiate, in case of an internet banking. What you do is that you say, look, Carol, this

was done from your account, you use the signature, you use the username, password and OTP and therefore it is you right? So that is how the banks handle repudiation cases.

But in case of blockchain as I said that blockchain is there is no central authority. So you cannot just go and complain to 10,000 people who are all part of the blockchain, they are all checking validity of transactions, they are all trying to make blocks of transactions and hoping that their block makes it into the blockchain. So, there is nobody particular to listen to your repudiation. So, therefore, this system should be non-repudiable. So any transaction that has been validated should be non-repudiable.

(Refer Slide Time: 37:46)



And this is where digital signatures come in handy. That is when Alice says that I am paying Dan 100 coins, Alice must sign it with her private key, and then Alice's public key should be available to everybody. So therefore anybody who wants to check the validity of the transaction, now, not only they check the amount of the transaction being less than the balance Alice has, that is one part of the validation. But the second part of the validation would be that they use Alice's public key to check it is indeed Alice's signature.

If it is Alice's signature, then only they declared the transaction is valid. Therefore, that transaction once it is there, becomes non repudiable because without Alice's private key nobody

could have signed for Alice. So therefore, the signature verification has to be done. Now if the signature does not check out, so they know Alice's public key and they check and the transaction is saying that, it is Alice is paying Bob, but is signed by somebody else, then we did not have any reason to validate that transaction.

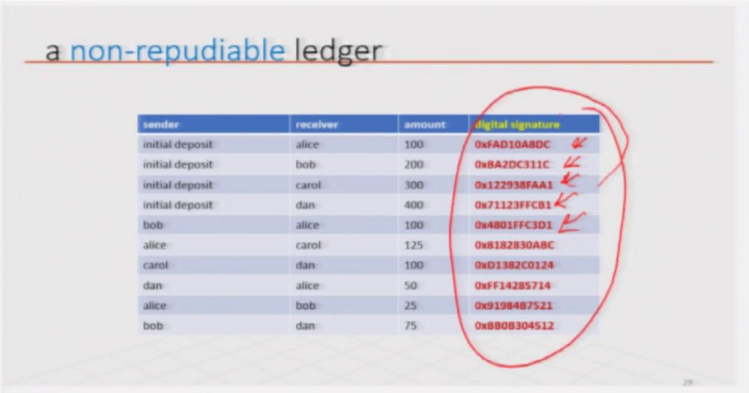
So therefore nobody should be able to forge signature without Alice's private key. That is the assumption now if Alice's private key gets stolen, then that is a different story. But we assume that Alice's private key is Alice's responsibility. And also, if Alice said that I want to pay 100. But somebody in the middle intercept, see this message is going to the broadcast to the entire blockchain. That is how the blockchain works. Everything is sent to everybody.

So if Alice pays Dan, let us say, somebody changes it to say that Alice pays Dan 200 coins, Alice said 100 coins and each sign the transaction, remember and how they sign the transaction, they basically take the transaction message and then apply the private key and then they encrypt it. And this encryption is basically with the private key of Alice. So with the public key, somebody should be able to open it and then they will get the message back.

So, therefore, if message is forged by somebody, they have to also forge Alice's signature. Otherwise, if the message is changed to message' when somebody encrypts it with some other private key, then that will not be able to check out with the Alice's public key right. So, the when you try to decrypt it, it will come out as garbage. So, therefore, if the transaction message comes through with Alice's signature.

And I am able to decrypt it with Alice's public key that means that Alice must have not only done this transaction, her transaction amount or any information she has put in the message is also not tampered with it so, that basically gives us the not only authenticity, also the integrity of the transaction.

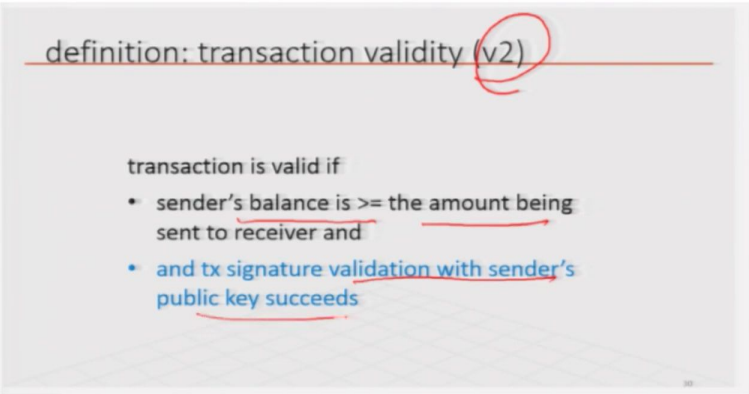
(Refer Slide Time: 40:51)



sender	receiver	amount	digital signature
initial deposit	alice	100	0xFAD10A8DC
initial deposit	bob	200	0x8A2DC311C
initial deposit	carol	300	0x122938FAA1
initial deposit	dan	400	0x71123FFCB1
bob	alice	100	0x4801FFC3D1
alice	carol	125	0x8182830A8C
carol	dan	100	0xD1382C0124
dan	alice	50	0xFF14285714
alice	bob	25	0x9198487521
bob	dan	75	0x8808304512

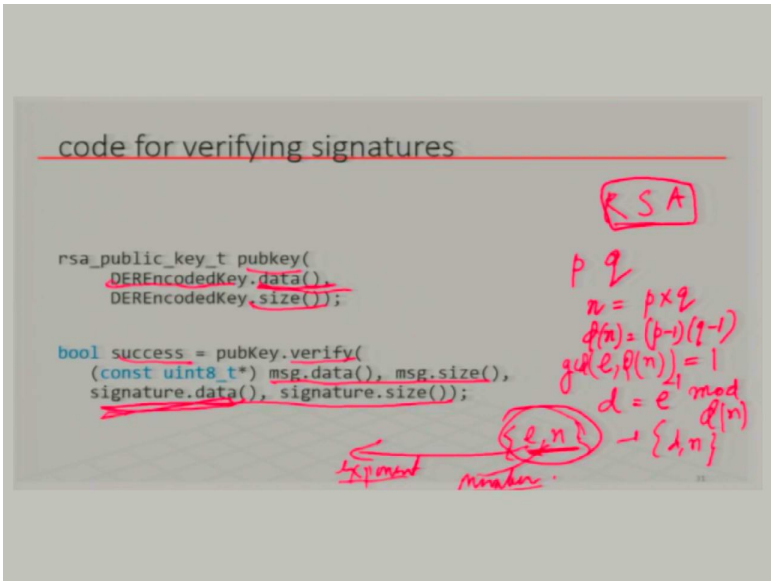
So, therefore, now, in the ledger, we have to not only store the transactions message or the information like who gets how much we have to also put the digital signature that came with it. So that later on if I again want to validate all transactions, I should be able to validate it by using the public key of the person who put the transaction in the sender and then use that to verify that the transaction amount and the whoever is saying the transaction is from are genuine. So, we have to also add the digital signature as part of the transaction information.

(Refer Slide Time: 41:30)



So now we have the transaction validity redefined or extended the definition. So not only the balance of the sender should be greater than the amount. Also transaction signature, validation succeeds. So the sender's public key should be used to validate the signature and it should succeed.

(Refer Slide Time: 41:52)



So now how do we do the signatures in programmatically? So there is this RSA public key type So, RSA we talked about the Rivest, Shamir, Adleman. So, here we are assuming that we are going to use RSA public key cryptography, and how a public key is represented is actually there are multiple different ways of representing a public key. Remember we said that in RSA as an aside, in the previous some previous lecture.

We said that we take 2 very large numbers prime numbers p and q, we compute

n = p * q

and we compute

phi(n) = (p – 1) * (q – 1)

Then we choose a number and then we choose a number **e whose GCD, with phi(n) is 1** and then we choose

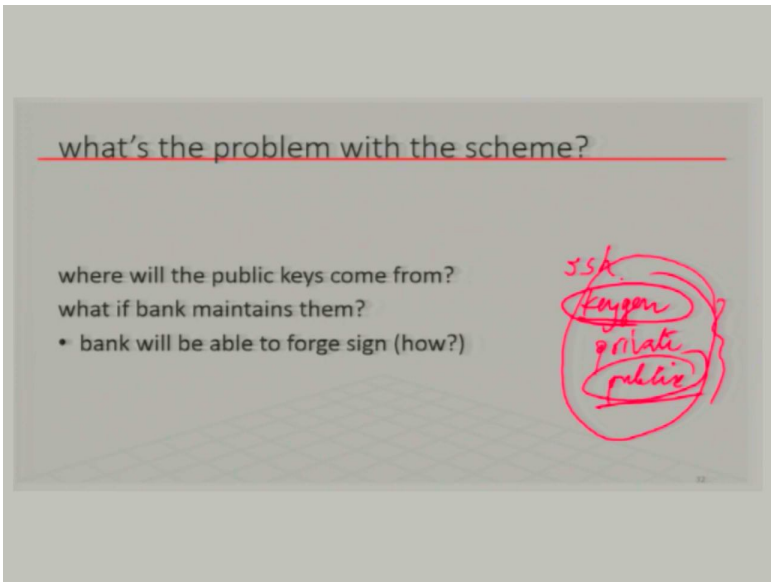
$$\mathbf{d = e^{-1} \bmod phi(n)}$$

and then we publish e, n as public key and we keep d and n as private key.

So, in this case, the public key is represented by the, **e**(this is called the exponent)and **n**(this is the number). So, this is e and n has to be represented and how do you represent that you actually do sometimes you do these are very large integers. So, if you will represent them in binary and then convert them to hex or you can represent them to binary and then you can do a base 64 encoding. So, based on that there are many encoding schemes, multiple encoding schemes.

So, here we are talking about the, DER encoding, which is one of the standard encoding of the key. So, the data of the public key will read **DEREncodedKey.data()** information and then the size of the public key will read the **DEREncodedKey.size()** information. Now, so you have this public key has a verify function and verify function basically is success when the signature, verifies with that public key. So the public key information is here, the signature information is here, you send it to the verify function. And if it is success, then we say that we have verified that the signature is genuine.

(Refer Slide Time: 44:33)



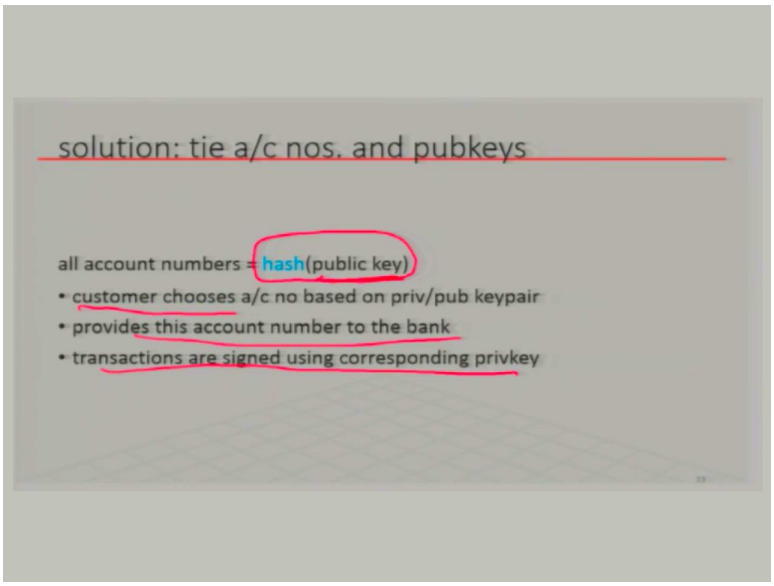
So, that is how the signature is going to be represented. Now, the question is, where do you get the public key? So, you know, in ssh, if you have used ssh, then there is a keygen function that allows you to create private key and public key, so when you use a key Gen function, that you can create your own private key and public key, then you can keep the private key guarded by some passcode. So that nobody else who accesses your computer cannot easily read it unless they know your passcode the public key.

You have to publish or you have to upload to let us say, github or something. So, so the so this is how we actually generate the keys. But that is not the point we are discussing here generation of the key is not difficult. What we are discussing here is what we discussed in a previous we also is, which is, how do you tell other people what your public key is? Now, in our setting, for example, in this particular example, we can tell the bank, our public key and the bank can store them.

But the question is, if that happens, then the bank can misuse them, especially if the bank has the private key also, like if the bank gives us a public key and private key, then bank generated the private key also, and therefore, the bank will know the private key for public key, so, then bank can actually forge messages transactions on our behalf, which is not good. So, similarly, if I have a blockchain which is not maintained by a bank.

But maintained by let us say a whole crowd like of 10,000 people who are all participating in the blockchain, then and I generate my private public key pair, then I can announce that this is my public key. Now, the problem with that is that what if somebody else announces a public key as mine. So, that problem is still there.

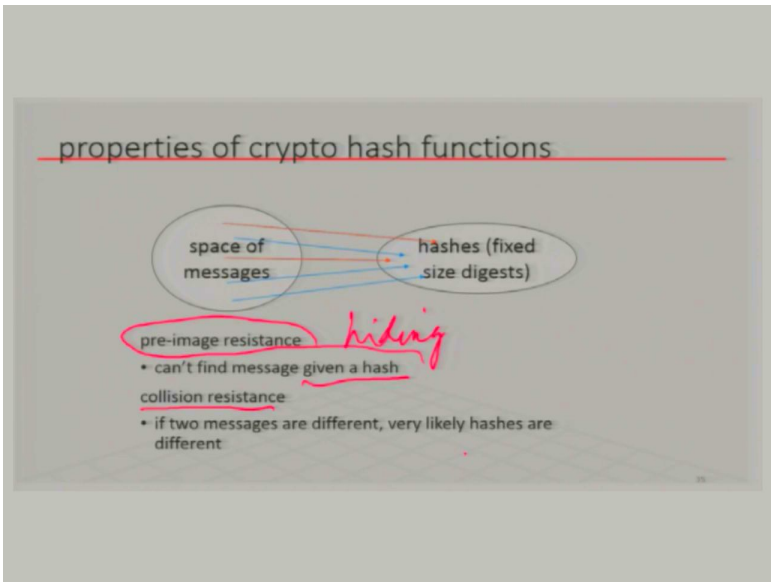
(Refer Slide Time: 46:40)



So, in our case, in our single controlled blockchain case, how we solve it here in this example, is that instead of bank giving us account number see right now, when you open an account in a bank tells you this is your account number. In this system, what we assume is that I generate my public key private key pair, keep the private key and hash the public key right and send bank the hash and say that this is my public key, this is my make this my account number.

So, the customer is now choosing his account number banks may or may not like that, but in this case that since this is a made up bank, we assume that the bank is okay with that. So, bank gets the account number for me. And when I sign transactions with the public with the private key, then my account number can be come handy in very validating that that my signature is genuine.

(Refer Slide Time: 47:50)

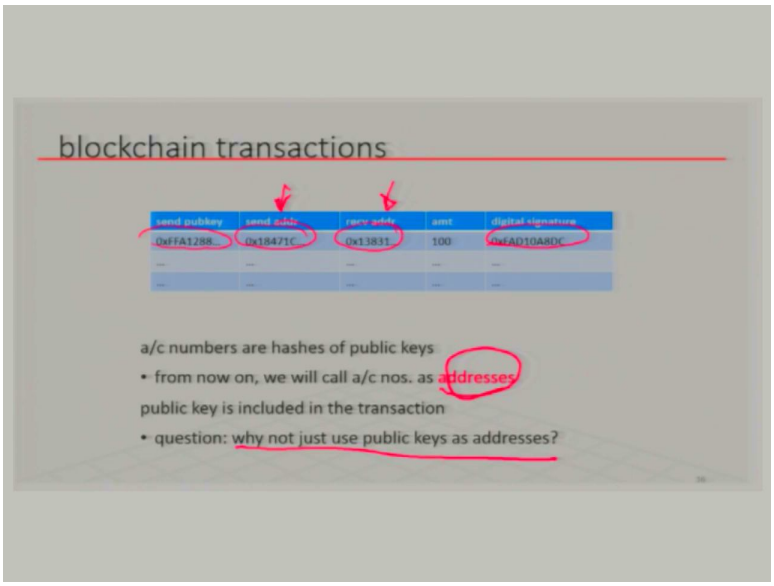


So why do we want to hash first of all, the public key if you generate a public key using key gen, you will see that the public key usually very, long string of numbers and therefore, nobody can remember or it is going to take like, you know, many many you know letters or characters length string to represent the account number, but if you hash it, then you can map it to a fixed length digest and then you are going to use that maybe 16 or 32 byte digest and then use that as your account number.

So, we will revisit some of the hash function properties, although we have already done this, that the number of real keys because the keys have much, larger size, if you hash them, many of the keys will become the same hash, the hash function should be such that that should not be common at all and not only that people should not be able to find that in case they want to and so, that see the 2 properties one is that hiding property or what is called this is the hiding property.

We talked about before that given a hash, you should not be able to find what the original was and then collision resistance that is, you cannot create 2 messages yourself whose hashes are going to be colliding.

(Refer Slide Time: 49:27)

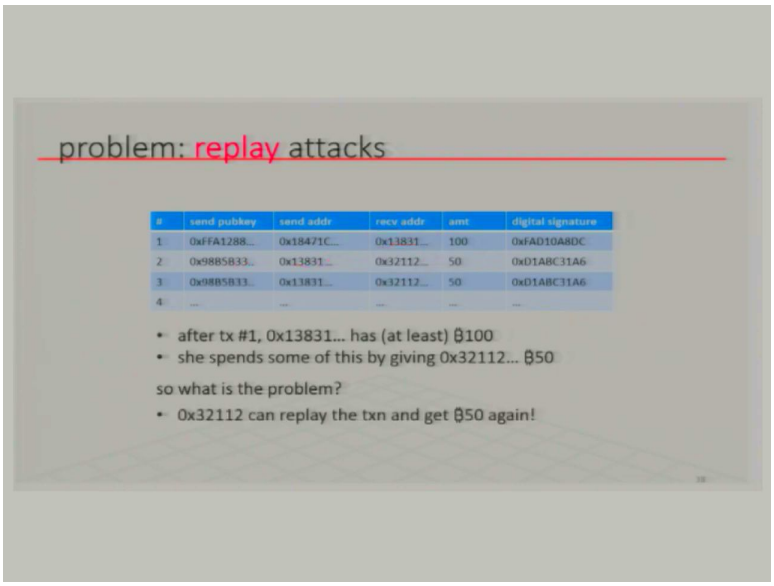


So, now, what you do is that a transaction will have sender's public key senders address, receiver address amount and a digital signature. So, the address is the hash of the public key. For receiver you only need the hash of the public key as the account number because that is the account number. Whereas for the sender, you need to also tell the public key because otherwise how would you use their digital signature.

So, this account numbers are also called addresses in line with how Bitcoin calls them. So in Bitcoin also, the addresses of each person is actually a hash of a public key. So each person who wants to participate in the Bitcoin ecosystem, they actually create this public private key pair, keep the private key, the public key is hashed. And then that is announced as the address of the account. And that is the also called a wallet. So that is the wallet address.

So public key is included in the transaction. And you can check whether the public key is genuine by checking the sender's address by hashing the public key. So the question is, and we already discussed this, that why are not we using public keys as addresses because public keys are too long and too cumbersome to deal with.

(Refer Slide Time: 50:52)

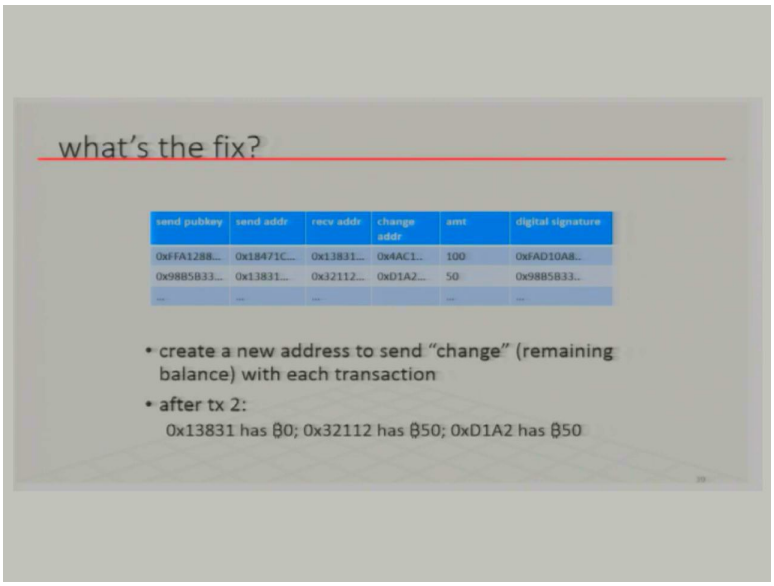


So now one problem here that can happen is called a replay attack. So let us say **0x18471C...** sending some 100 amount to **0x13831...**. And then **0x13831...**, now out of this 100 sends an amount 50. So after transaction one, **0x18471C...** address has 100. And then to buy something she uses 50. Now, somebody who is listening to this transaction actually can change this and for example **0x32112...** person can broadcast these transactions. Transactions are broadcast.

Because everybody needs to see all the transactions because everybody is keeping track of the transactions into blocks and so on. So the **0x32112...** person then broadcast another message with the same with all the same thing like the public key of the sender, the sender's address, his or her own address, and under 50 and nothing changes because message integrity cannot be sending the same thing again. So there is no message integrity problem.

So now, if others believe that this is another 50 coin transaction, then at the end of that **0x13831...** address will now have 0, although, she intended to actually spend only 50. So that is somebody replaying an old transaction again. So, this is one problem that one has to worry about. So therefore, what you do is you have to make some tricky solution.

(Refer Slide Time: 52:32)

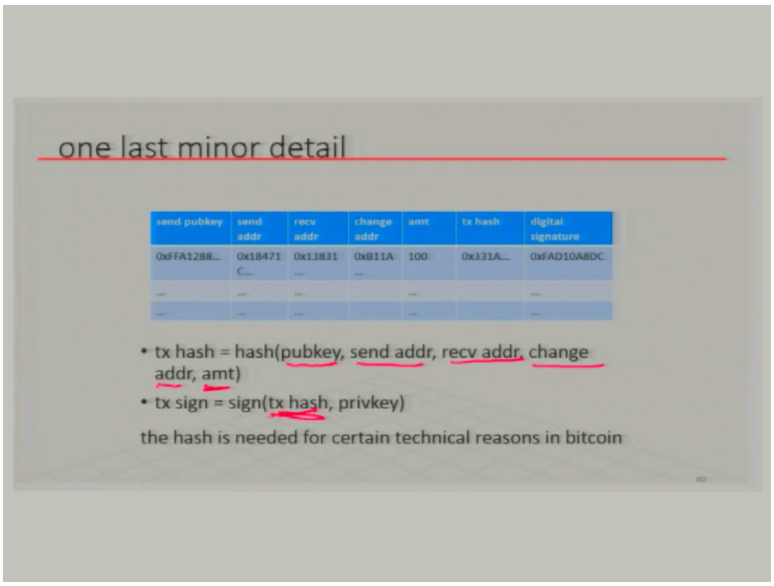


So every time you make a transaction in this kind of setup, you did not keep the balance or whatever the rest of the balance in the same account. So you create another account with a different address. And then the part of the transaction you say, that give 50 to this address, and give the rest of the other like, in this case, 50 to a new address. So now the original address no longer has 50 left original address will have 0 left.

Whereas, the receiver of the transaction will get 50 and rest of the 50 will come to my own a different address or to call an alias address. So, therefore, replaying the transaction will not help because when somebody tries to replay this transaction, the sending address no longer has any money. So, this is something that is very common, I mean, that is done in Bitcoin that every time you make a transaction.

You broadcast a transaction, if it is the full amount that is your only 100 and you are giving out 100, then that is fine, then in that case, you did not have to do anything. But if you are giving part of the, what you have, like in this case 50 out of 100, then the rest 50 has to go to a new address, it cannot remain in the old address because replay attack can happen. So this is something important to understand.

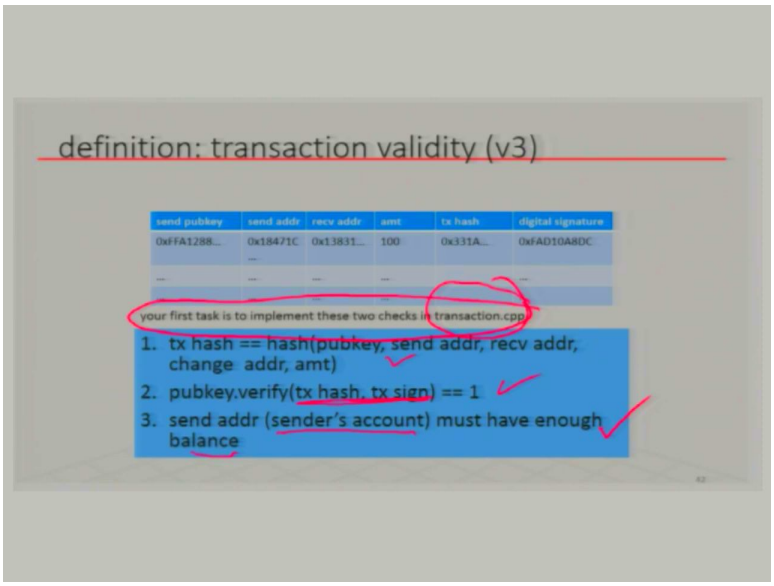
(Refer Slide Time: 53:59)



So one last minor detail for every transaction, we also add the transaction hash. Now remember the transaction contains the public key, the sender address the receiver address, the change address that is the new address to which you want to send the change left after the transaction and the amount of the transaction. So this is hashed. And then this hash can prove that your transaction is not tampered with.

And then you sign the transaction on the transaction hash rather than the entire transaction. Of course, that makes the signing easier because the entire transaction has a lot of information. And then if you create a digest, and then encrypt the digest with your private key that will be faster. So therefore it makes sense also to do the hash before you do the transaction signing.

(Refer Slide Time: 54:57)



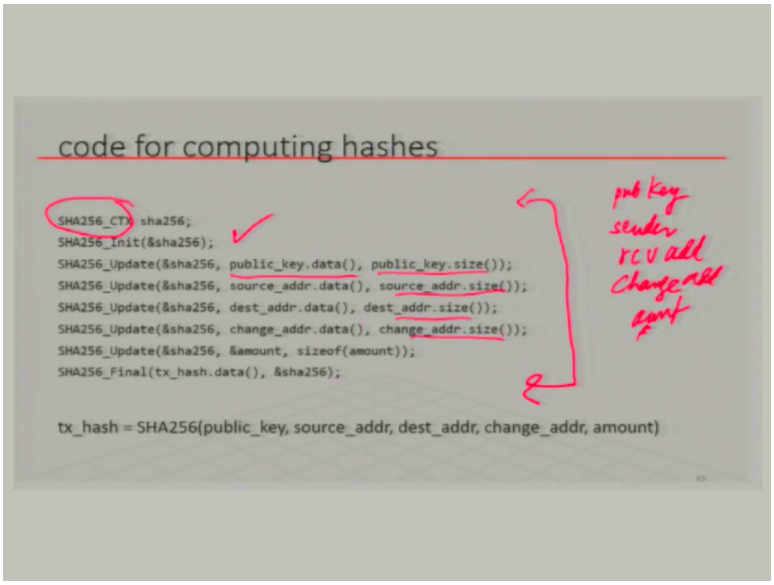
So the final transaction structure will then be a public key, source address, destination address, change address, amount, transaction hash and transaction signature. Remember that source address is result of a hash. Similarly, destination is also a result of a hash. And change addresses result of a hash all addresses are hash of the public key. But you also need to give the public key so you have to use the public key as a vector.

And then at the end, you will actually add another vector of the signature of the transaction. So this is basically c it is a sequence of bytes. Because it is a 8 bit unsigned integer where they are basically byte so you represent the public key as a sequence of bytes. Similarly, use represent the signature also as a sequence of bytes. So now we have a third definition of transaction validity. First, the transaction hash is there then you should be able to verify the signature on the hash. And then your sender's account must have enough balance.

So this we already had and we also had this signature thing, but the new thing is that we have to take the hash of the transaction and then verify the signature on the hash in the exercise that we are proposing. So we will give you a set of already almost implemented blockchain as part of today's discussion. However, the there are places where you have to fill the gap, otherwise, this will not work. So there is this function, this file called transaction.cpp, where you have to

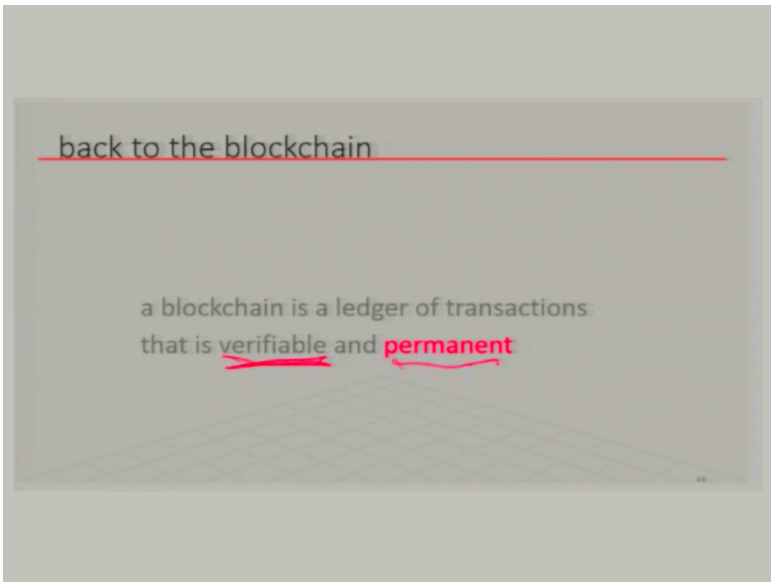
implement these checks, write the checks of the signature verification, and take up the balance of being enough.

(Refer Slide Time: 56:49)



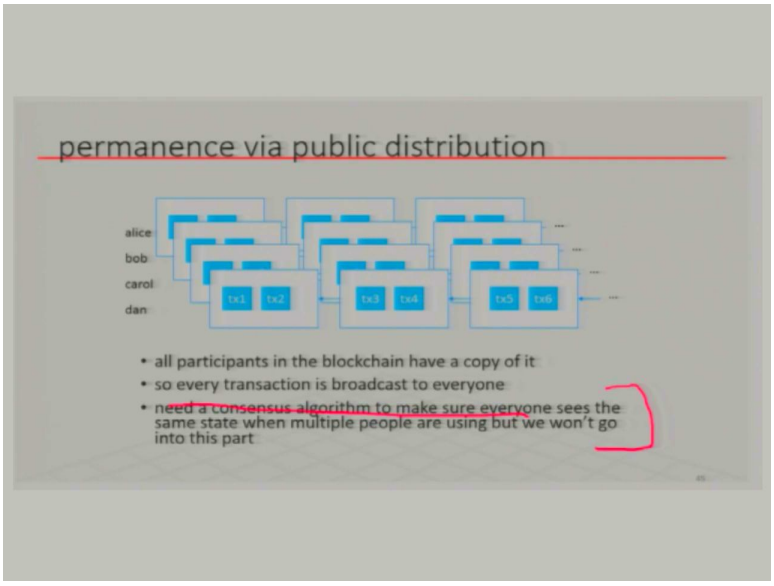
So this is the code for computing hashes. So this is a little bit airy, but this is the type of the SHA 256 hashing. So you initialize it and then you keep adding, what is the next piece of data that you want to add into the hash your data structure has this public key, you have the sender address, you have the receiver address, you have the change address, you have the amount and you have the signature, while signature is not part of the hash. So, what you are doing is that you are telling the size also and then you are adding one by one and updating the SHA hashing process. And eventually what you get is the transaction hash.

(Refer Slide Time: 57:37)



Back to the discussion on blockchain. Again, that verification part is done the question of permanence.

(Refer Slide Time: 57:45)



So, now, one way to have the permanence is that it is a highly tolerant to failure of storage. So, for example, if the transactions are all stored in 1 central location, and that location fails, then you are going to have you have lost the data? But blockchain is, one thing that happens is that chain is, as I said, is a dynamic structure. It keeps growing. But at any point, everybody has a copy of the current blockchain.

Now, they may differ slightly from each other because as I said before, because of network delay, and everybody is far from each other, my copy may be slightly behind your copy. I have not gotten the latest block added to my copy yet, but we have an eventual consistency that eventually everybody will have the same blockchain. So all participants having a copy of it. So we have the permanence in the sense.

That if I lose 1 copy, or 3 copies, I still have a lot of copies around. Second is that every transaction is broadcast to everyone. And that is how everybody independently creates the chain. So this is an interesting thing is that nobody is telling them what goes into the blockchain what is the next block, when a next block is created, multiple people will create next block. And I discussed before that this multiple people created next block is actually going to be slightly different from each other.

Because the last transactions they have seen their choices of transactions that they put into the block, all that stuff will define what each individual creates this block. Then they compete once they compete. And then some one of them will solve the hash puzzle first. And then he will declare himself winner and then he will broadcast the block with the proof of his win to everybody. So everybody is now getting a copy of the block that has one.

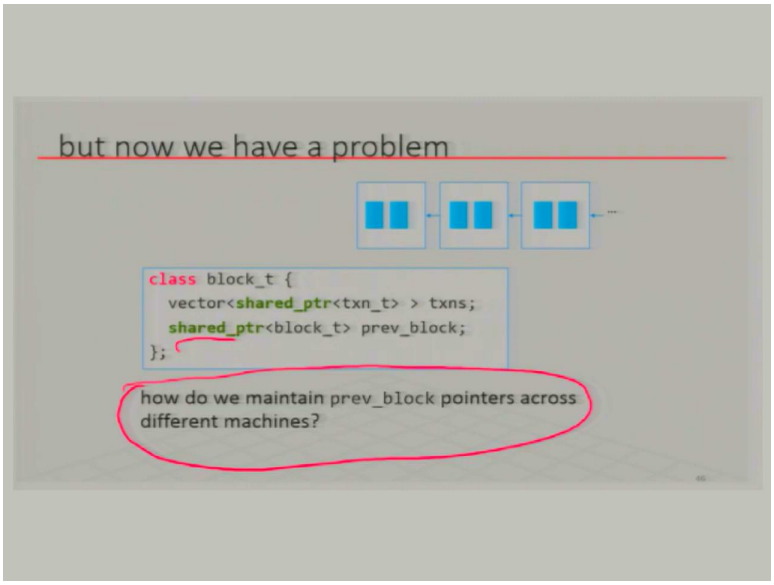
So even though I have I have been sitting there, by myself and I have created my own block also but unfortunately, I did not win. So once a winner block is broadcast to everybody, everybody adds that copy of that block to the latest blockchain. So that is how we distributed, eventually everybody has the same chain. Now there are some race conditions and some periods of time when multiple copies may differ from each other.

But we will see when we discuss Bitcoin blockchain, that eventually everything becomes convergent again. So there is a, transience that may actually make some difference, but overall, it works. But in our case, actually, that is not a problem because we have only one copy of the

blockchain that is being maintained by our program, but that is not real situation. So what I described as you know, this block, being winning, and then broadcast everybody, this winning process is called a consensus process.

And so there has to be a consensus algorithm. So what the hash puzzle solving is the consensus process, that is followed by Bitcoin and Ethereum, and few other blockchains algorand, for example, whereas many other block chains, there are other consensus algorithms, and we will see them like Byzantine fault tolerant consensus algorithms of various flavors. So, consensus algorithm is not necessarily always by hash puzzle solving. But the first few blockchains that we will see that is ethereum and Bitcoin will see that the consensus algorithm is indeed based on hash puzzle solving and therefore, based on mining.

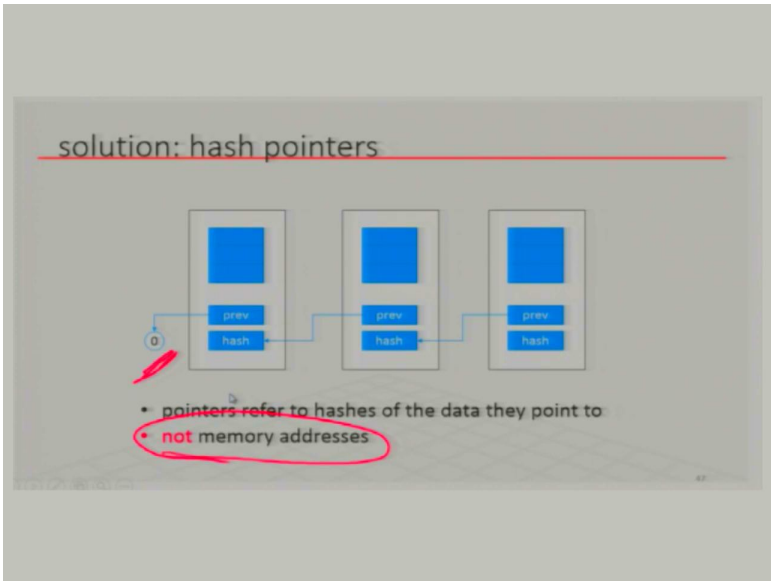
(Refer Slide Time: 01:01:57)



Now, as at this, one, this is where we actually are having an advantage in our particular example programmatic example is that we are only keeping one block copy of the chain in one within one program. So, we are able to create like shared pointers, which are memory addresses and so on. But in case of real blockchain we have the copies all across and in there on different systems they might be on Linux system, they might be on Windows system and so on.

So, therefore, the pointer sizes will be different based on 64 bit machine 32 bit machine and so on. So, therefore, this kind of pointers are not what is used in the real blockchain.

(Refer Slide Time: 01:02:46)



So, the previous pointer actually across different machines work slightly differently. So, the previous pointer in this case refers to the hash of the previous block. So, as I said in a previous lectures also, that, that is enough actually if you think about it that I have many blocks, I have the hash of some block, which I considered my previous block. So all I have to do to check which one is my previous block, I have to go through all the blocks and compute their hashes.

And whichever guy's hash matches with the hash that I am carrying as my pointer is my previous blocks. So this way I can keep building the, the previous to previous and so on. And I can actually cache it. So at some point, I will not want to, every time I want to find who is my previous block, I did not want to do all the hashing and computation, I might because once the blocks have become permanent, their hashes also become permanent, so I can actually cache that information.

But the point here is that the having the hash of previous block in yours stored inside your block is good enough for finding the previous block and that is how the chain of blocks happen and the first block or the genesis block does not point to anybody before it because there is none before.

In the real blockchain, they are not memory addresses. They are actually hashes and they are called hash pointers.

(Refer Slide Time: 01:04:11)

The last thing that where do the bitcoins come from so the bitcoins are created whenever a block is mined, that is when a block is created, everybody tries to solve the consensus problem and can discuss in case of Bitcoin hash puzzle, and then as a reward of the winner with a winner is given some new bitcoins and that only way Bitcoin is created. So the reason why reward is there is that because otherwise, why would I spend my computing resources, which requests money to validate transactions to put them in a block.

And then solve a hash puzzle to be part of the blockchain? I need some reason to do it. So there has to be an incentive. So this incentive or reward actually makes the system keep going otherwise everybody will stop mining because it is expensive to mine. So, the reward should be also enough so that the amount of money they spend in paying for electricity bill and paying for the hardware and paying for the cooling of the very large hardware that they use to solve the hash puzzles should be surpassed by the reward.

So the reward should be big enough and this is called a block reward. So the block reward initially in Bitcoin was 50 bitcoins, then after 4 years, it became 25. And after 4 years, it has now

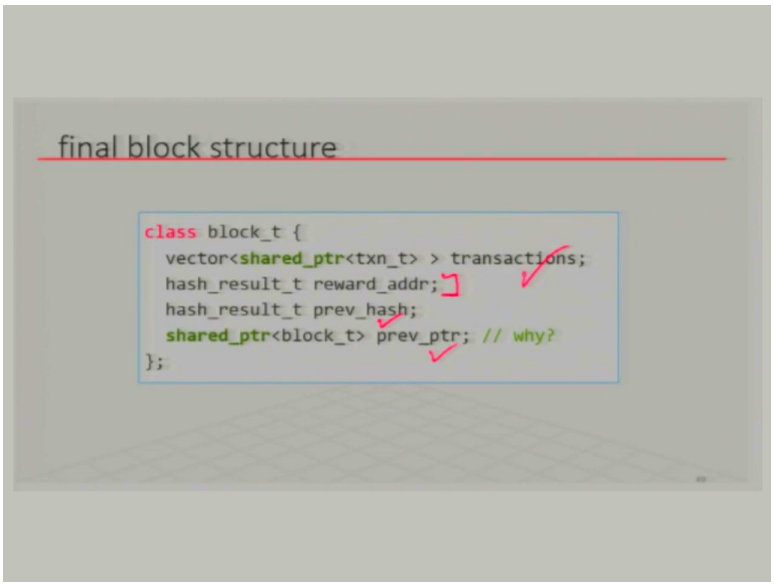
become 12.5. And in another couple of years, it will become 6.75 and so on, and at the end, it will become 0. And the reason is that the Bitcoin ecosystem also has this thing that there will never be more than 21 million bitcoins unless the Bitcoin changes its strategy and policy and everything.

So and the only way to create bitcoins is through the block rewards. So as we, if we keep the block reward the same, always, then the, then you have to start, you know, past the 21 million mark. So therefore, they are tapering it off, it is like kind of like a geometric distribution so that the eventual sum becomes 21 million. So then you might ask, why would, somebody be interested in doing mining, spending, you know, money to do the mining solving hash puzzles.

So there is something also that every transaction that you make, you are also has to give some transaction fee. So in the beginning, the transaction fee was in there, and we have not included in our example here. But if the transaction fees there, which means that whoever wins them hash puzzle can keep the can transfer the transaction fee from each transaction to his account. So therefore the, if there, you know, 1000 transactions in a block, and each gives.

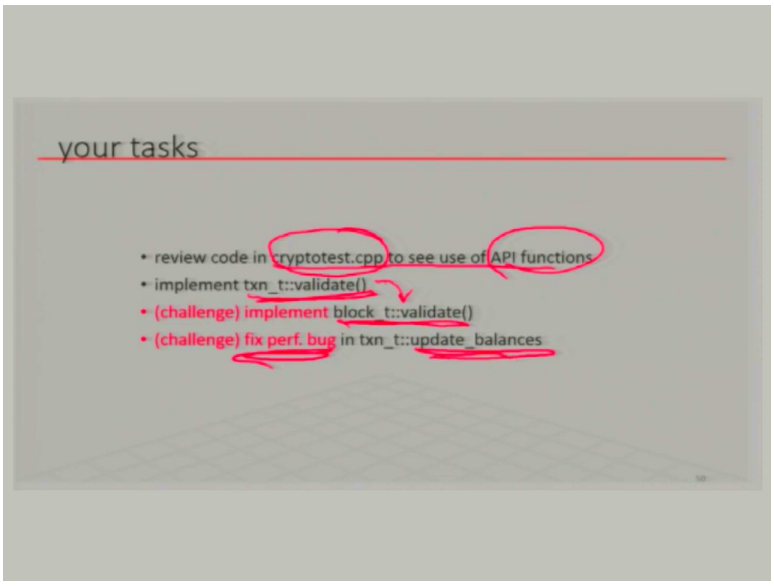
Let us say, some small amount of transaction fees, then together, they will become a pretty good amount, and therefore, the miners will still have incentive to gain new bitcoins through this process even though the rewards have stopped. So this is how this thing is planned to happen.

(Refer Slide Time: 01:07:34)



So the final block structure is that you need to add also the reward address. Because whenever there is a reward, in the block we should know where to send the reward. Rest is the same as before. It is a list of transactions, previous hash, and in our case, it is a previous pointer also because we are doing it inside the program otherwise that party's not necessary to be there.

(Refer Slide Time: 01:08:08)



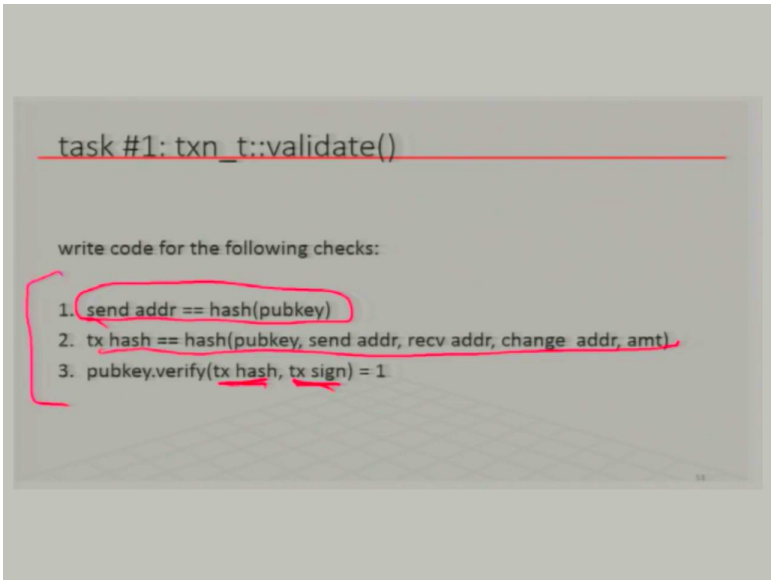
So, now coming to the, your optional exercise based on this particular lecture. So, first of all, we will give you a zip file containing the code and it will have some instructions you have to actually use ubuntu linux to do this. So, you if you did not have ubuntu linux, you can actually

use virtual box which you can download from oracle and virtual box is actually a virtualization software and then you can download an Ubuntu Linux which is free.

And then you can have a built a virtual machine using Virtual box which runs Ubuntu 18.4 or letter, because we will also will need you to get some of the libraries that are not usually coming with the Linux distribution. And those libraries are working only with the latest Ubuntu. And then you look at the code in this file crypto test.cpp to see how the various API functions from this libraries that that you have to use for example, for generating keys and for hashing, SHA256 and so on.

And then you have to implement this validate function. So, the final version 3 validate of transaction and then you have to implement the block validation that is how to implement the validation of all the transactions by calling this function actually you can validate and then you can also, there is a performance bug in the code that is given in the update balances function. So that you can find then you will have some good exercise in you know performance debugging.

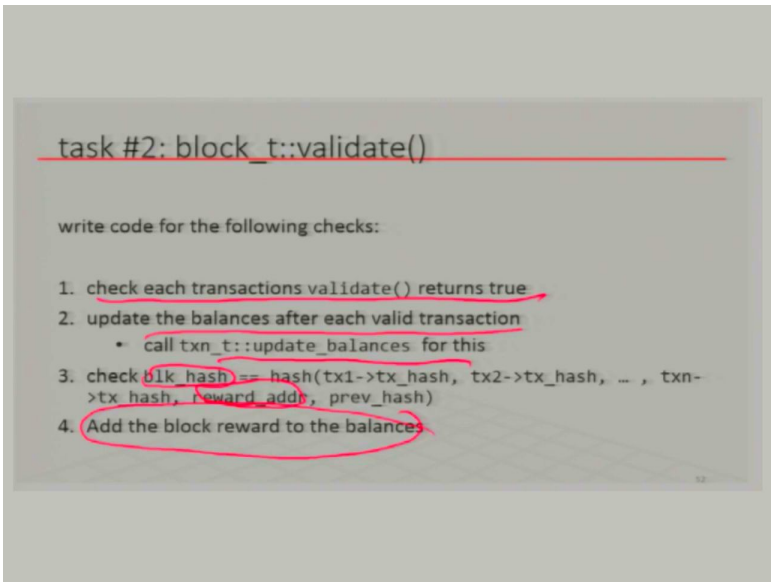
(Refer Slide Time: 01:10:08)



So, for transaction validate, you have to write code for the checking that the send address is the hash of the public key that is part of the transaction also you have to compute the transaction hash and check that the hash that that has been put part of the structure matches and then you

will also have to use this transaction hash and check the signature that is there. So, these are the things that you have to check for validating a transaction other than the balance check which actually we are assuming.

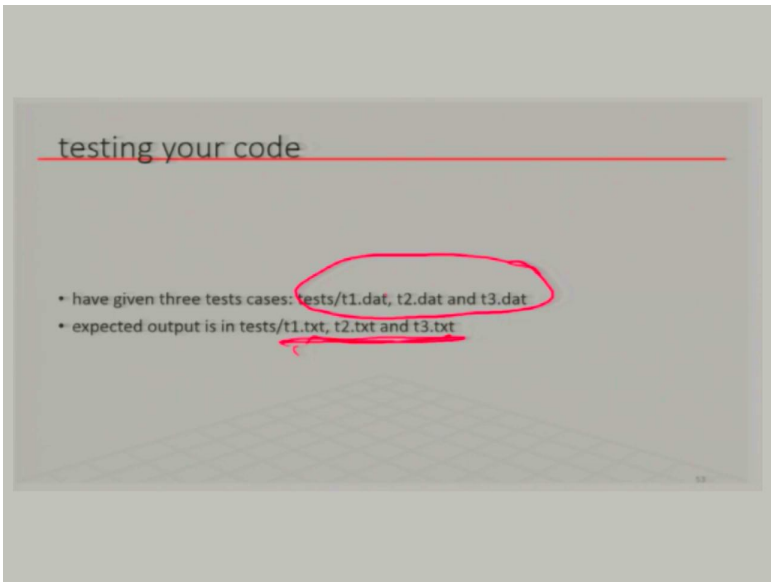
(Refer Slide Time: 01:10:45)



You will do for block validate, you have to go through the see block is a vector of transactions. So, you have to iterate over the vector and apply the transaction validate to each of those, and if any of them returned false then the block is invalid. If all of them returns true, then the block is valid, then you do update the balances after each valid transaction. So you can call existing update balances for this, then you check the block hash.

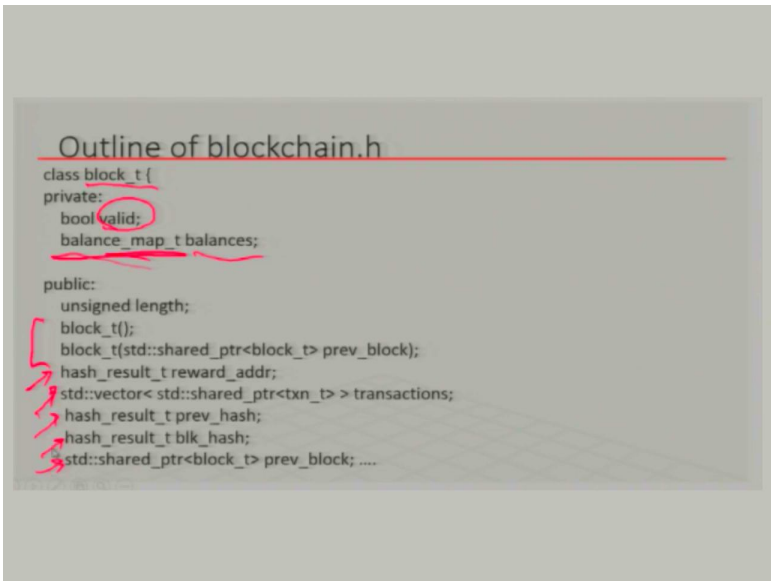
And then you add the block reward to the balances, wherever in this case, whatever the reward addresses, you just put it there, in case of a distributed computational situation like in bitcoin, every person who creates a block puts his own addresses the reward address, but eventually only one of them will get reward.

(Refer Slide Time: 01:11:38)



Now in the same zip file, you will get a number of test files, number of data files, and if you use them, then you should get the output so you should match them against these outputs by doing a comparison of the this file against the output of this running your program on this data and that will give you a satisfaction of having done it correctly.

(Refer Slide Time: 01:12:02)

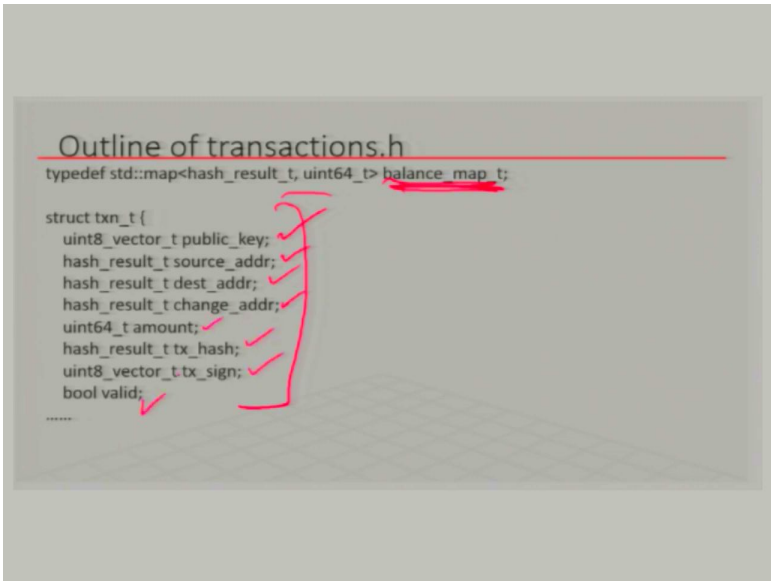


So this shows the outline of the classes. So earlier when we were actually going through this, we had lesser, you know, we only showed the snippets. So this shows a little more like you know, this is the block class whether it is valid or not, after the validation process goes through, it will

be set to true or false. And then you have the, remember this is a map, this map is a hash table for balances. And then you have all these functions like these are constructed functions.

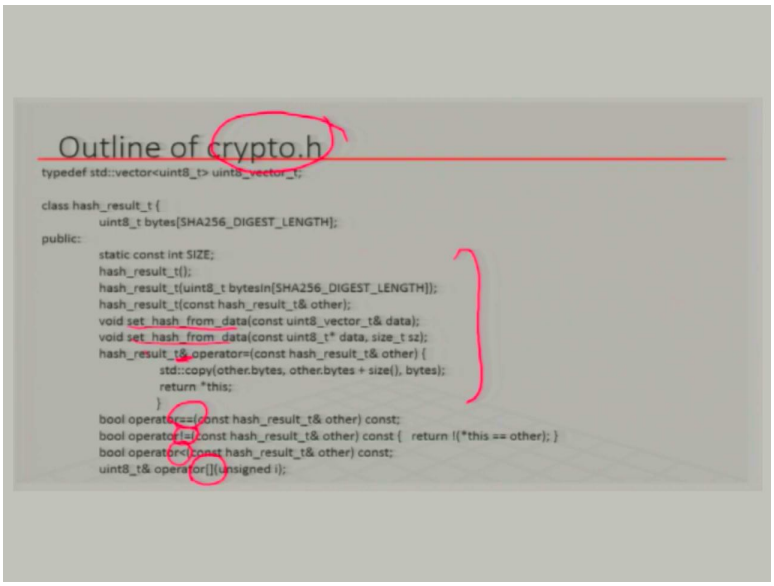
This is actually a data item. This is also a data item, which is a set of transactions as a vector, this is the previous hash, the block hash, and then this pointer is there for our program, because our program is based on a single process. Therefore, we can use real pointers.

(Refer Slide Time: 01:12:53)



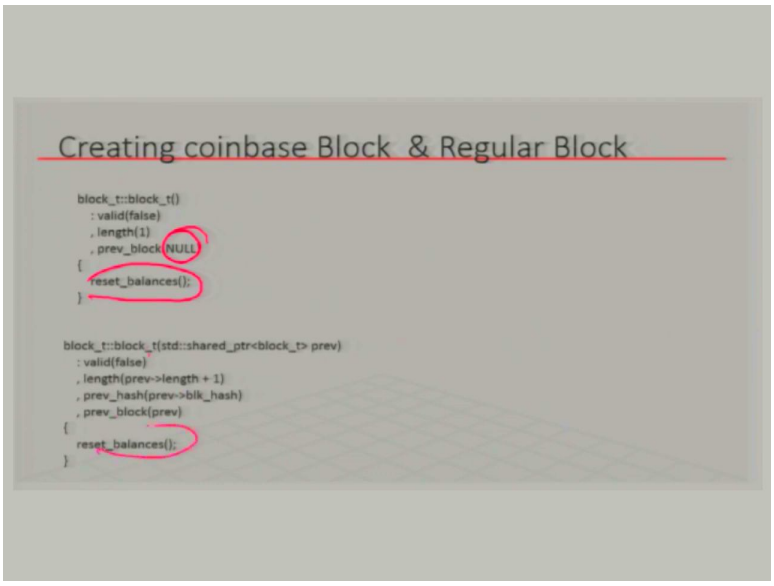
Transactions are also actually having this structure and this balance map hash table type is actually defined in this transactions.h and this is what are the different parts content of the transaction we already saw have the public key the source address, destination address, change address, amount, hash, sign and whether transaction is valid.

(Refer Slide Time: 01:13:20)



Hash result is the so, the crypto.h has prototypes of the crypto functions and the classes in this case the hash result class is defined here and it has all the constructors, it has the way to set hash from data and also it has a hash result. There is the **&** operator overloading. There is the == overloading, there is **!=** overloading , there is **<** overloading. So, these are the kind of things and this is how you access a particular part of the hash results. So, not all of them are required for this particular program, but this is how the hash result class is defined.

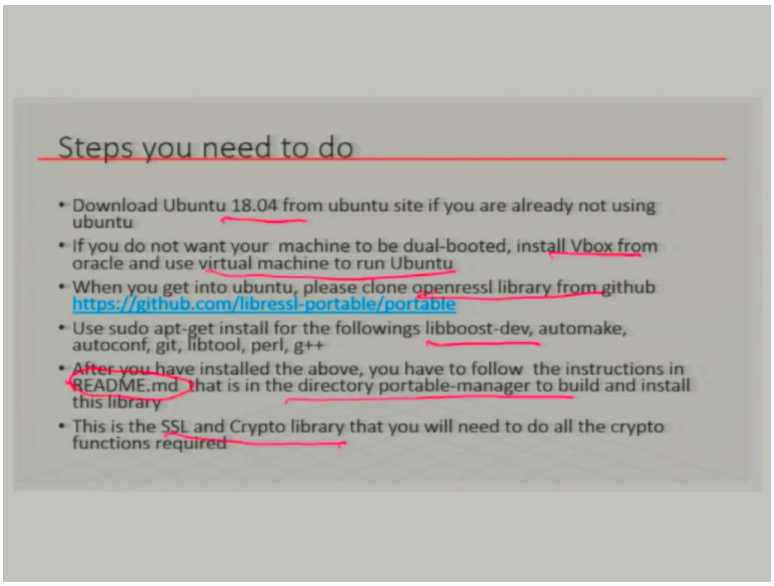
(Refer Slide Time: 01:14:08)



So, the first block is called also called the coinbase block. So, coinbase block can be created using this constructor. And when you say that the previous block points to null and it's length is one it is validity is set to false initially, later on if it is validated, then it will be done and in the beginning you reset balances for regular block initially you create you initialize all these different data items like the validity is set to false length is updated.

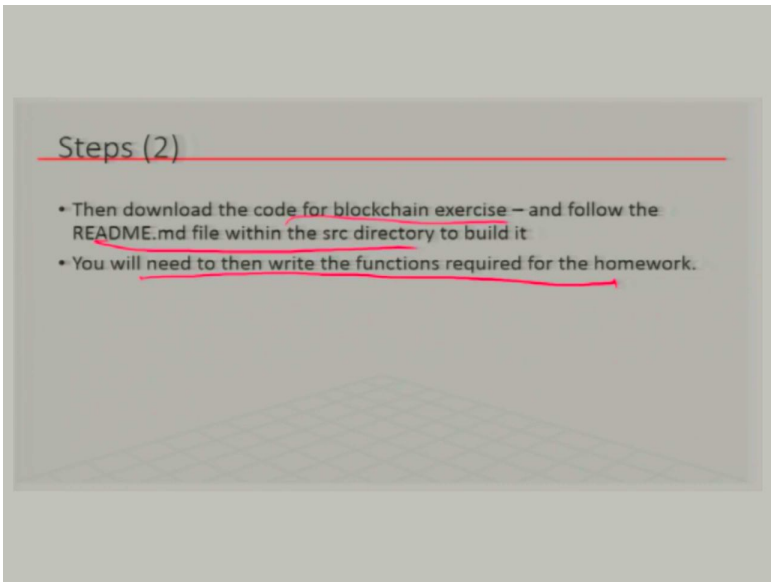
To the previous length plus one previous hash is the previous blocks hash and previous block is you know the pointer to the previous and you reset balances after that with different functions you keep adding transactions and so on in the block.

(Refer Slide Time: 01:14:58)



So, the steps you need to do is you have to download the latest Ubuntu. And if your machine is not dual booted, install Virtual box from Oracle and use virtual machine to run Ubuntu. Then you go to clone the open ressl library from github. And then you do the installation of the various libraries boost library, automatic auto configure, etc. Once you have installed all this, then there is a README.md file, which you have to view to see how to build and install the portable manager library. And this is the where you get all the crypto functions that are required.

(Refer Slide Time: 01:15:42)



And then you download the code for blockchain exercise from the sherm platform that we will be putting in there and follow the README file within the sociality to build it. And then and then you write the specific functions that you require in the homework. So the writing the functions is not a lot of work here. But what is happening, what is going to happen is that you have to understand the entire code to actually make the changes, small changes.

And that is where you will get to see the entire code and understand how it works and so on, which is the goal of this exercise. So with that, this lecture 3 comes to an end. So, next time we will actually look at the first real blockchain you know, how it is done. You know, it is, you know, techniques and structure and you know, how the distributed maintenance of it is done for Bitcoin blockchain. And we will go from there thank you.

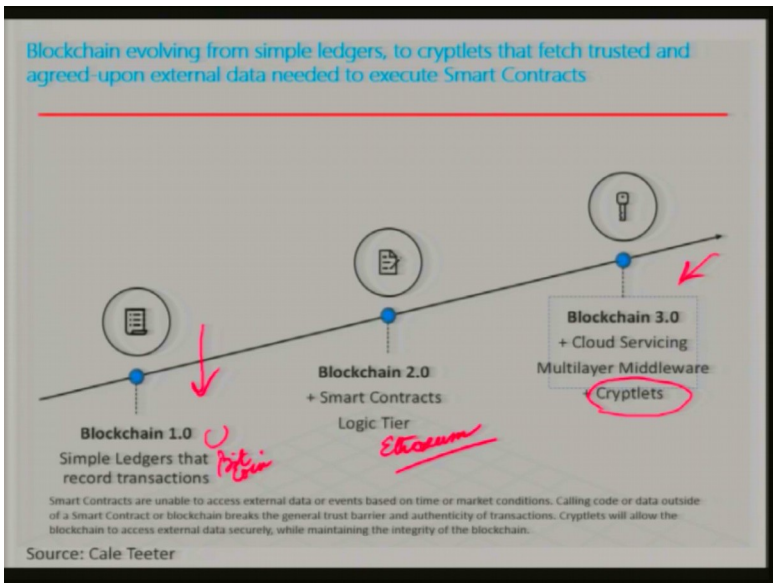
Introduction to Blockchain Technology and Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture No. 04
Blockchain Technology and Applications

Welcome to another session of blockchain technology and applications class. What we are going to do today is dive deep into the Bitcoin blockchain. In the beginning, I said that this course is not about crypto currency. But then why are we talking about Bitcoin blockchain? I have indicated before that Bitcoin is the first application of the blockchain technology, and therefore, it introduced a number of different technologies working together.

So it is more like integration of lots of existing concepts from cryptography from distributed computing and data structures. So therefore, it is no blockchain discussion can be complete without the discussion of how the Bitcoin blockchain works.

(Refer Slide Time: 01:10)



Also, if you look at this picture here, the blockchain technology is also evolving. So, what is happening today is that we have come a long way from the 2009 version of blockchain which was introduced with bitcoin. So, we normally call that as blockchain 1.0. So, blockchain 1.0 is basically simple, Ledger's that record transactions. And the idea of that recording of transactions was to make sure that any transaction that you make you ensure that it is replicated in many places, and also it is collected into blocks.

And then the blocks are made permanent by creating hashes of blocks and then storing the hash in the next block, and also making each subsequent block somehow dependent on the previous all the previous blocks by this hash chaining. So that is the blockchain 1.0. Then blockchain 2.0 was introduced with the idea of smart contracts. So, when we go and look into ethereum, later in this class, you will see that the blockchain 1.0 did not have a way of programming, how you want to use the blockchain.

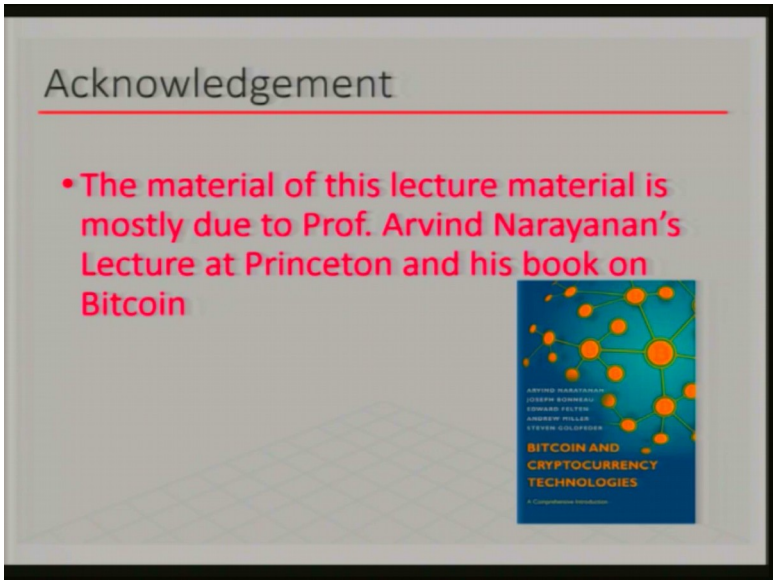
So, you can now in blockchain 2.0, you can actually write applications on top of blockchain that uses the basic primitives of the blockchain to accomplish data storage, transactions, and many other functions with smart contracts. So, that makes the blockchain 2.0 an example of that early example of this was ethereum and early example, of blockchain 1.0 was Bitcoin now we are moving into blockchain 3.0 so blockchain 3.0 is actually with the blockchain with smart contracts.

But it also has cloud, multi layer middleware, and also something called cryptlets. Now, we will go into that much later in the course, if we have time, but the idea is that neither blockchain 1.0 or 2.0 can interact with the real time data that is coming from the from the outside. So, there are some tricks that are used in blockchain 2.0 to actually bring in real time data for example, stock prices that changes in real time.

Now in within, you know, 6 to 10 microseconds and that data every time you want to validate something, that data has changed. So therefore, there is a problem connecting real world data to the blockchain, especially for function, like a smart contract depends on what is the current stock price, for example, that kind of things might actually be a little problematic, and therefore, cryptlets are somehow designed to have a secure way of interacting with external data.

So, that is the evolution of blockchain technology. And that is where we kind of stand today. So therefore, what we want to do now is focus on blockchain 1.0 for now, and understand how it works, and then we will move on to blockchain 2.0 and then finally, we will see some of the features of blockchain 3.0.

(Refer Slide Time: 04:47)



So today's lecture is actually based on Professor Arvind Narayanan from Princeton, his book and his lectures in at Princeton on a course on Bitcoin. So we will selectively use some of the ideas and some of the material that he used for his course. So we should acknowledge him.

(Refer Slide Time: 05:07)

So the first thing that we have to understand is the question of centralization versus decentralization. And we discussed this before that you have a way of collecting information like transactions, let us say in your bank account, and keep them in a one database. Or you can also have one replica of the database for fault tolerance or for backup, but the point is that the entire data is stored in a central location or in within the authority of a central, trusted body, like your bank.

Now, that obviously has its own problem first of all, centralizing everything gives you this notion of, you know, vulnerability in the sense that if that particular data is somehow lost by a cyber attack or some other way, then you are going to lose all the information. Second is that the central authority has full command over what can be done on the data. And you have to somehow trust that central authority not to manipulate the data or delete some entries or do something like that.

So, for example, Aadhar is a perfect example of a centralized database. So all your biometric data, and everything is in central data's data centers. Some of them are in Noida, some of them are in Hyderabad. But the point is that if you have your Aadhar number and biometric, then people actually use API's to access that data to authenticate you that is through the KYC process, when you use your Aadhar based KYC.

Now, if the central authority maintaining the database decides to delete your entry, then you no longer have any access, nobody will be able to access your data and therefore, you will be not recognized through the KYC process. Now, unless there is a alternative way of doing KYC, you will be in trouble in the sense of a sense that it is not it is denied that you exist. So, so that is a problem of the centralization, too much power in the hand in the hands of a central authority.

So, therefore, now, Satoshi Nakamoto he was very worried about or he or she or them, they were very worried about these centralization of monetary infrastructure and monetary information in the hands of few banks and 2008 there was a big banking crisis in the US and, a lot of banks actually went bankrupt and government had to bail them out because they were too big to fail.

And therefore, he was saying that we have to deliver ourselves from the tyranny of the banks, and then we should actually create a currency that is completely decentralized. So, there is no central authority which creates that money or a central authority, which, keeps the keeps track of the money or authenticate people to use their money. So that is the basic, so, you know, political economic underpinning of the introduction of the Bitcoin blockchain.

So centralization and decentralization versus decentralization is a basic concept that underlies the blockchain technology. Now, centralization has many advantages.

(Refer Slide Time: 09:13)

For example, it is easy to manage. So a bank will not have to worry about, you know, having replicas and privacy issues. Because if you have many replicas, you have to make sure that the data is kept private, all that kind of stuff. It is easy to provision. So for example, if you want to create a new account, you have to just make change in the central database. If you want to delete an account or then also you have to change in one place, it is easy to ban.

So for example, if you have a centralized authority, maintaining for example, your DNS domain name service, then you can easily ban certain domains from being found, and it is easy to distribute responsibility. So centralization has all these advantages. However, decentralization has some of these disadvantages, for example, it is harder to manage if data is distributed all over the place.

It is harder to distribute work in the sense that you cannot say, you know, you have this responsibility, and you have that responsibility, because nobody is being commanded by a central server or central authority. And it is harder to ban something so you cannot delete information so easily. And it is harder to provision. But centralization, as a single trusted party, is its biggest weakness in terms of not only the fact that they might actually do things in a unilateral way, which is not good for the consumers.

But also the fact that if it gets cyber attack or any kind of attack, then also it can actually have a devastating consequences. Whereas if the data is decentralized, and all the functions are decentralised then it is much harder for an attack or for a central authority to you know take

over dysfunctionalities or change the functionalities and so on. So, therefore, blockchain is based on the notion of trust.

So, the question is when you have a central authority, you have to trust that authority and normally in the past we have been doing that. So, we have been trusting the authority that is managing the DNS, we are trusting the authority that manages the digital certificates, we are trusting the authority that is managing our accounts. But in the current time, we have been, we have we are seeing that a lot of the times the central authority becomes overly authoritative, they ban things.

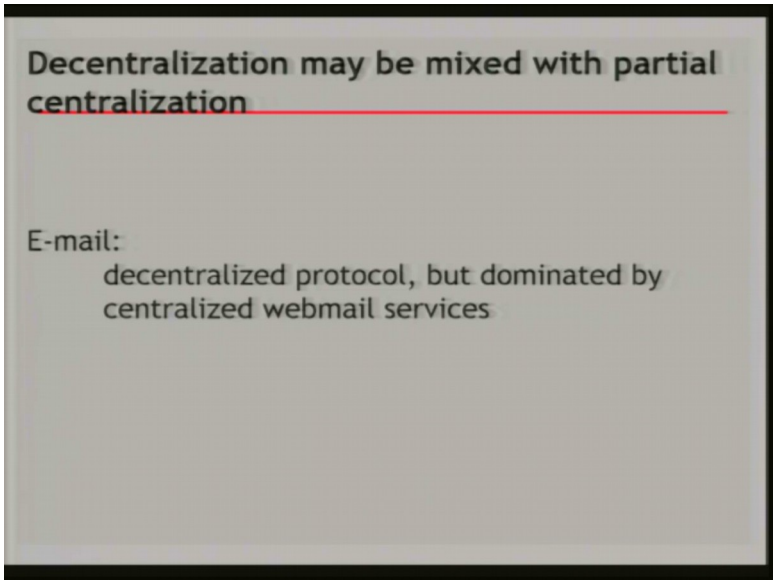
They actually can change things that to put you into trouble so therefore, the trust has to be created from a decentralized system. Now, in a decentralized system when you have many players and no player is more important than the other players, which is the case in the centralized situation, there, creating trust is also a challenging task. Because you have you may have a number of the players in the system a number of actors in the system, who are also malicious.

So, you have to somehow assume that in your system in the ecosystem who are actually together doing the computation together, keeping track of data together deciding what is valid and what is not valid. You have to assume that some of them will actually do something maliciously and so, therefore, your system should be designed in such a way so that in spite of that you have the ability to trust the system.

So you have deriving trust from untrusted participants or an untrusted actors for that you may have to make certain assumptions. For example, what percentage of people are untrustworthy? And what percentage of people are trustworthy if that assumption sometimes have to be met in order to trust the entire system. If you did not trust anybody, 100% of the people, then you cannot build a trusted system out of 100% untrusted actors.

So we will see how all these assumptions play out in the blockchain, especially in today's class, we will talk about bitcoins blockchains trust how do we how we derive trust in the system. Now sometimes, you know, it is not a black and white thing.

(Refer Slide Time: 14:02)



So we may have decentralization, but we may also have partial decentralization or partial centralization. For example, email system, the SMTP protocol is a completely decentralized protocol. So we can have SMTP servers. And the email client that we use are SMTP clients. So you have all these clients. And we have, we can have many, many SMTP servers who talk to each other. So if my SMTP, I send an email, my SMTP server then talks to the other SMTP server in the destination.

And then that SMTP server will actually forward the mail to the SMTP client that is the let us say, outlook or whatever, to read that email. So this is a completely decentralized protocol. It was designed like that however, in the past 20 years or so, almost 30 years now, since the introduction have Hotmail, Yahoo Mail, and then now Google Mail, and Microsoft mail and so on, we see that certain all mails eventually go to the SMTP server of those 4 or 5, 6 different web mail servers.

And then from there, everybody gets the, everybody is received that email. So I your, if you are using Gmail, for example, and your sender is also using Gmail, then they both are talking to a single SMTP server, and not like their local SMTP server talking to the local SMTP server of the receiver. So that is a decentralized protocol that is designed, but for convenience, and for better management people have found kind of a centralized way of managing the entire email infrastructure.

So this can happen other centralized infrastructure or, for example, when you buy a digital certificate, you buy it from a Certification Authority. So there is a usually a Certification

Authority, which is, which may be, affiliated to a larger Certification Authority, but in general that Certification Authority issues and signs that digital certificates so when you buy the digital certificate you are from a digital certificate issuing authority, then what you get is a digital certificate signed by that authority.

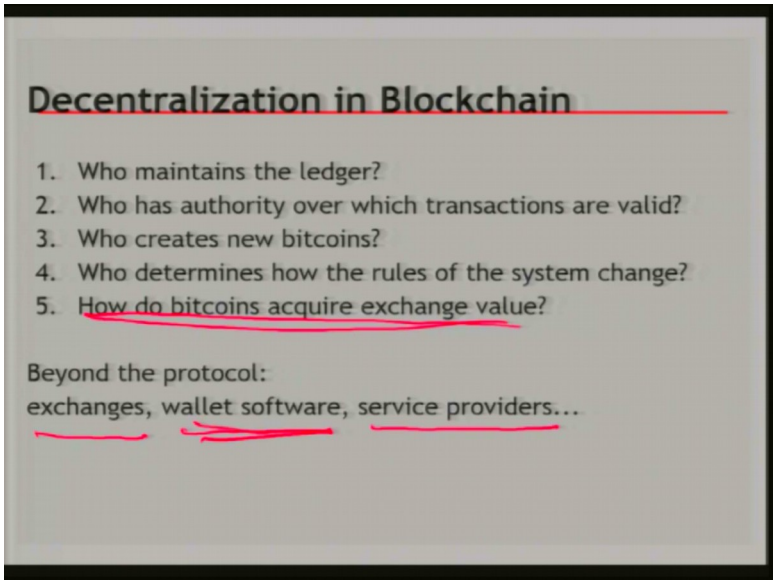
And then that authority has to be trusted by the person who is actually going to use your digital certificate to, for example, send you encrypted message or check your signature, things like that. Now, if that or authority, then becomes annoyed with you and then decide to revoke your certificate, then you can no longer use your digital certificate because the other entities will not be trusting that that digital certificate that you have because it has been revoked.

So here again, the central authority will have too much power to actually control whether you are trusted by others, or you are able to digitally sign documents, things like that. The other central example centralized example is DNS. So DNS is the domain name server. So domain name server like when you have a URL, for example, let us say www.google.com. Your browser then asks the DNS server who is, what is the IP address, for that for that domain name because the internet protocols did not understand domain names.

They only understand numeric IP addresses. So it goes to the DNS server. It has a hierarchy which is there is local DNS server there is regional DNS server, there is a domain, top level domain DNS server and so on. But eventually it goes all the way to a central root server. So in this case also suppose somebody decides, the companies that manage this DNS servers decides that google.com is no longer a desirable domain.

If the authorities that maintain the DNS servers decide that google.com is no longer a trusted, trustworthy domain name, then they can revoke it, and therefore, the Google will have a problem. So, these are the kind of things that can happen in a centralized server based system. So, that is why the decentralization comes into picture.

(Refer Slide Time: 18:52)



Now, how is blockchain decentralized? So let us ask a few questions. And if you can answer these questions, then you will know whether it is centralized or decentralized, for example, who will maintain the ledger. The ledger is basically the transaction records. Now, as we saw in the previous exercise that we talked about before, that we created a ledger, which has all the ingredients of a blockchain except that the ledger was maintained by the program that you were writing. So there was a centralized ledger, even though other components of the blockchain was there.

However, if you do that, then that centralized ledger, centralized program, which is maintaining that ledger will have all the power to change things and stuff like that. So therefore, in blockchain, we dos not maintain the ledger in a central only one location. It is replicated among 1000 and 1000 of different computers and different what we call nodes. Now who has authority over which transactions are valid?

So, in a regular banking situation, you see that the transaction validity is checked by the bank in the program that we have done as an exercise. There also my program or the program you will write was taking the validity of the program validity of transactions. Now, we said that the validity of the transaction has this property that amount in the balance has to be greater than the transaction amount greater than or equal to the transaction amount.

We said that the detail signature must check out all this stuff but if it is only one authority that is checking it, they might surreptitiously change the logic they might say, I will allow now transactions which has which is which will end up in negative balance from the sender's

account or I will not allow this person signature because I did not like this person. So, centralized authority for transaction validity checking is also not good.

So, in blockchain, actually, every participant who is interested in data mining or keeping the ledger is going to check the transaction validity. So, if somebody wants to act maliciously, but others will not necessarily act maliciously, at least, the majority will not act maliciously we assume. And therefore, the transaction validity should be within the logic that we have come up with. Now who creates new bitcoins?

So we know in a banking system, the new currencies created by the central authority like Reserve Bank of India, or Federal Reserve in the case of in the United States, but in the bitcoin blockchain, the bitcoins are not created by any central authority. In fact, we discussed before that whoever actually wins the competition to create the new block. And if that block is actually accepted as the new block, then they get some reward.

And that is the only way to create a new bitcoins. So therefore, it could be anybody who wins and not every, time same person wins. Therefore, bitcoins are created by many, participants. Now, who determines the rules of the system change? So bitcoin has now certain system rules that everybody is supposed to follow every participant. So that is how they program the program they write to run the mining process or check for transaction validity is based on certain rules.

Now, how do you change the rule? The question is that you might want some rule change because you found some deficiency in the system. And that is when the rule changes. When rule changes the current bitcoin current part of the blockchain that so far you have developed may become obsolete. So, you may have to fork the blockchain. So, that is also not done centrally because this forking process takes place kind of automatically through the entire dynamics of the system.

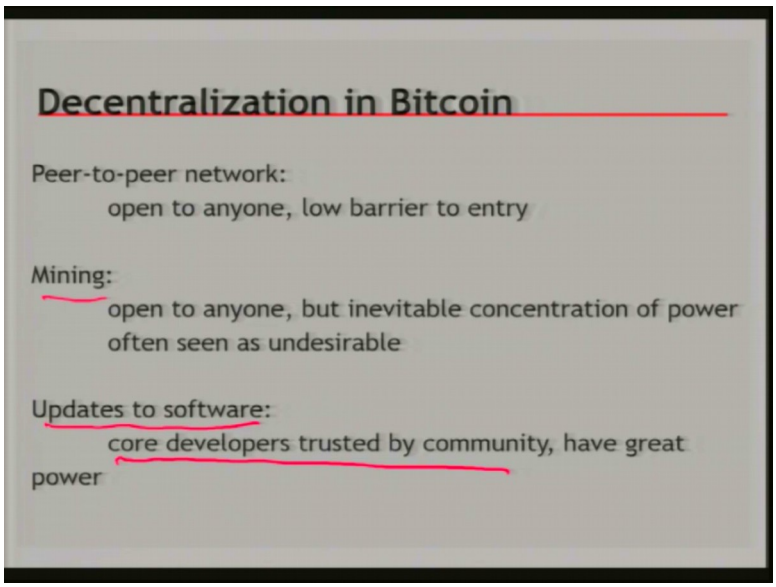
Now, other question is how do the bitcoins acquire exchange value that is, what is the monetary value the fiat currency value of bitcoin? So, last time I checked, I think, this week, the bitcoin prices about is between $7000 and $8,000 per bitcoin, there was a time when it was only a few 100 dollars, and there was a time when it almost touched 20,000. Now, who decides this exchange value there is no central authority here.

But this exchange value is not part of the blockchain dynamics or the blockchain programming or anything it is done external to the bitcoin blockchain it is done by the market, but the market works also in a decentralized fashion because more people if more people get interested in the bitcoin buying bitcoin, then its price goes up, if people start feeling that bitcoin is may have a problem.

Then for example, when news comes that China is going to ban bitcoin or things like that, then the picture bitcoin prices go down. So this entire going up and down is based on how many people are trying to buy bitcoin or how many people are trying to sell bitcoin. If more people want to sell bitcoin, then the price will go down. So that is how this process works. So these are actually this thing is actually beyond the blockchain protocol.

This is actually completely extra external to the blockchain protocol. So the bitcoin exchanges, the wallet software, some wallets software is the software that you if you are participating in the bitcoin in buying and or doing transaction with bitcoin, then you have a wallet software, you have various service providers. So these are external to the protocol. So we will not worry about these in this course.

(Refer Slide Time: 25:19)



So how is decentralization achieved in bitcoin? First of all, bitcoin actually works with a peer to peer protocol. So if you make a transaction, you actually want to tell the entire ecosystem that I am making this transaction. And here is my signature authorizing this transaction. And

this is broadcast, how do you broadcast it, you broadcast it by sending it to all your local neighbors, and then all your local neighbors, then will send it forward to their neighbors.

And this is how the blockchain transactions are broadcast throughout the entire network. And this entire network is actually overlay network over the TCP/IP internet and so on. So, therefore, in this network, the nodes are whoever is participating in the bitcoin ecosystem. And the communication is done through an overlay over the TCP/IP because the bitcoin endpoints are basically applications, programs that basically do all the work.

So, anybody who wants to say that I want to join the bitcoin network, he can just have the software that is required to join the this peer to peer network and he can enter and the way he enters he has to tell what is this public key, actually the hash of the public key as we discussed before, as an address, so, he says that here is my address, and here I am. And here is my public key. So then the rest of the world eventually would know that there is another such node that is from this part of the network.

Now, the mining process that is mining is a process of deciding, which is the next block that will be added to the blockchain. Even though the 10s of 1000s of copies of the blockchain at every node, there is 1 copy, but the copy is dynamic so, because it keeps growing. So, what gets added next has to be agreed on by all these different entities because otherwise the blockchain will be looking different at the different nodes.

So, how this process works is through what you call mining, and anybody who wants to do mining can do it so, if you want to do mining, you can also do it. However, it turns out that today, doing a bitcoin mining requires you to solve a very difficult hash puzzle. And we saw before that solving a hash puzzle requires brute force computing and therefore, you need a huge amount of computational power and only a few players seem to have acquired that much computational power.

So, with a single laptop or a single workstation, you cannot really ever win the hash puzzle contest. So, therefore, you will never be actually mining any block. But if you have millions of dollars worth of equipments and software, then you can sometimes win the blockchain this hash puzzle and therefore, you will be mining and you will get rewards, the question is, who gets rewards?

See, third thing is that the updates to the bitcoin software, sometimes you may decide that the rules have to change and we are there you find a bug in the software, which everybody else is running. So, therefore, there are some core developers, which are who are trusted by the community and they have to announce a new update, everybody has to get that update and if the update is very, you know, changes the way validation has to be done or mining has to be done.

Then there will be a big change in the blockchain. And maybe sometimes blockchain may fork. So we will stop here. And then when we come back, we will we will actually go into the idea of consensus mechanism in bitcoin blockchain.

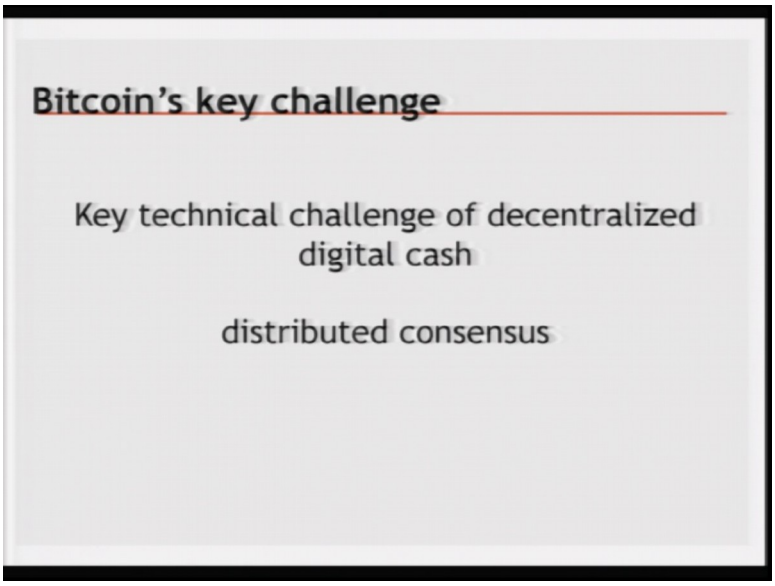
Introduction to Blockchain Technology and Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture No. 05
Blockchain Technology and Applications

(Refer Slide Time: 00:14)

Welcome back. So, now as I said before that we will be discussing consensus mechanism in Bitcoin blockchain.

(Refer Slide Time: 00:22)



The main technical challenge, so digital cash has been thought about and many proposals have been made by computer scientists for almost 20 years before the 2008/9, when the Bitcoin blockchain was introduced, and that was the first cryptocurrency that we saw. So always there was this question of the consensus that bothered people. Now why is it important because when you create new money, or when you create a new transaction?

Somebody has to validate that this is the valid transaction. Somebody also has to check that you are not double spending so you acquired some money. So let us say 25 bitcoins, and you pay somebody 25 bitcoins and again, you pay the same 25 bitcoins to somebody else that should be disallowed. Now, with a central authority that is very easy to check, but with a decentralized system, it is much harder problem.

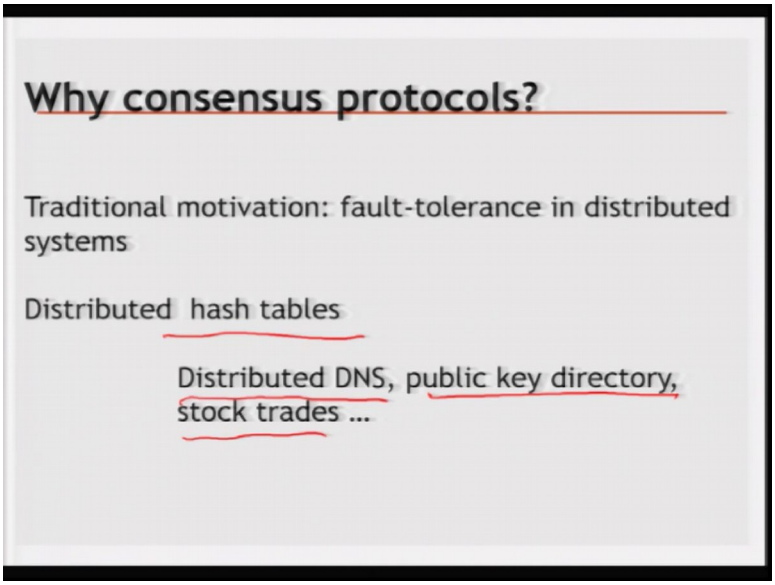
Because if one node here says that what you are doing is not acceptable, other nodes may say that what you are doing is acceptable. So, therefore, one has to decide that what should be the verdict of the entire network. Because somebody might be maliciously saying that what you are doing is not acceptable when you actually did perfectly correct thing. In another possibility is that the some of the good participants got fooled.

Because you did something wrong, but you actually fooled them or it could be that the number of malicious guys are going trying to say some bad transaction is good transaction like you have

done double spending and then you have told your friends that you vote for me and you say that this is a good transaction that transaction gets into the blockchain and it becomes permanent and that is not good.

So, therefore, this consensus is a mechanism by which you say we or all of us agree to agree on something or that something is acceptable. So, in a distributed setting, consensus has always been a problem. And this problem has been looked into almost for 40 years.

(Refer Slide Time: 02:57)



So, let us see what are the things that happens; now consensus as I said other than in case of blockchain how the consensus problem came about in the in the distributed computing community. So, in a distributed computing system you have distributed set of nodes and this set of nodes have to compute something. So, let us say they have to compute some function or they might have to compute that you know, some routing table or they may have to compute, they may have to together maintain information and information integrity.

These are the kind of things that distributed computing community worry about now to in order to have a distributed system work, many times people do what is called a leader election. So you have n number of nodes, and then you run a protocol that everybody engages in and eventually say, here is the node that will be the leader. So once I have done the leader election, I will listen to the leader throughout the computation.

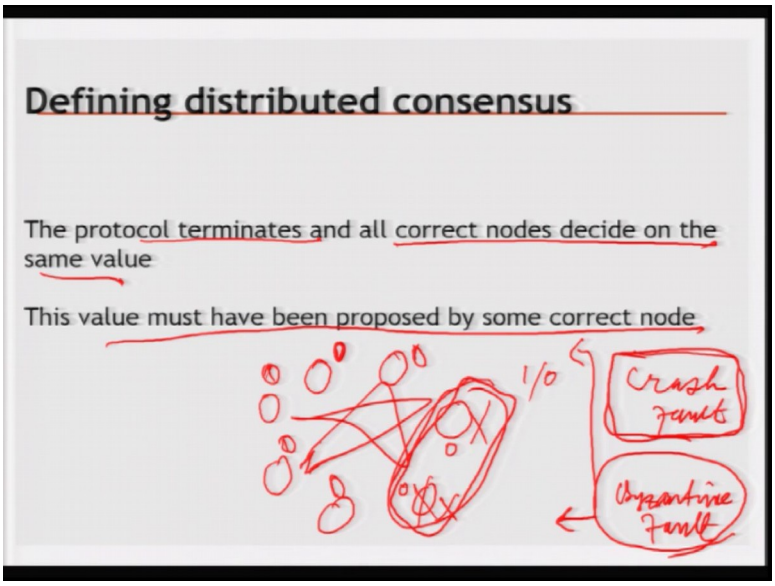
So that leader can then give you work that you do this amount of work you do this amount of work, and together you accomplish something. So that is basically called a leader election protocol. So similarly, be sometimes the distributed system has to do some voting. For example, for leader election also, you may have a voting based leader election. But sometimes you have to vote on a piece of data. And therefore, you may have to do something called a consensus.

So this problem has bothered distributed computing people for more than 40 years now. And some of the applications they had thought about even before blockchain is that I may have to create a hash table, but I did not want to be wanted to be maintained by just a single authority. I want this hash table maintained by everybody. So everybody will have a copy of the hash table. And this copy of the hash table should be the same.

If you have that, then you can have a distributed domain name service. Earlier I said that the distributed domain name service right now is kind of centralized. But if you have a distributed DNS, then you know, you are free from the dependence on a trusted party public key directory. So when you have public key remember like yesterday, we discussed that if you have public key, you know, telling everybody else that is it your public key.

And also convincing them that it is your public key requests digital certificates. Digital certificate is basically when a trusted third party tells the others that indeed it is your public key. Now, you can actually maintain a public key directory distributedly in a decentralized fashion through a distributed hash table if you can solve that problem. And then other places like stock trades and so on. So there this has been a problem distributed computing problem for a long time.

(Refer Slide Time: 06:06)



So how do we define what the distributed consensus problem in a formal way, so what we say is that distributed consensus is a protocol that terminates and then all nodes that are not malicious decide on the same value and this value must be proposed by at least one correct node. So, let us say you have n nodes, nodes means these are computing nodes and they are connected by some topology here. And they are going to decide on some value.

So, everybody proposes a value for example, if it is a binary consensus, then everybody proposes either 1 or 0. So, everybody proposes something. And then you run the protocol at the end of the protocol, everybody should have, let us say among these, these 2 are bad factors. And they are good, they are proposing 0. Everybody else is proposing 1. Now, during the protocol exercise, these guys will not also listen to follow the rules of the protocol.

So they will try to send messages, they might try to confuse the others. But if the distributed consensus protocol is robust enough, that is, with the existence of this malicious guys doing malicious things. Eventually, all the good guys will decide on one because one was something that was proposed by one of them actually, in this case, in my example, all of them, but maybe some guy might have done something else.

But what they agree on is 1, then we know that we have reached a consensus. Now since one of the good guys also did 0. In this case, if everybody eventually agrees of 0 that would be also a

correct outcome, because what it says is that the protocol terminates and all correct nodes decide on the same value and that value must have been proposed by some correct node. Now, the only complication here is that protocol designer does not know how many are going to be malicious node here.

Second thing he does not know is who are the malicious ones right. So therefore, the protocol has to be designed in such a way so that the no matter who are the malicious nodes, this consensus will eventually terminate and a consensus will be met and the consented value by the good nodes. Non malicious nodes should be chosen from the values that were given by atleast one of them. So anything that the malicious guys proposed as the value that should be chosen.

If none of the other good guys have proposed that value, then that should never be the end result of the consensus. So that is the idea. Now, it turns out that depending on what kind of maliciousness you are going to see, in these nodes, the problems gets different. So for example, in the very beginning, people started saying that, well, if the while the protocol is running, if one of the nodes crashes, so that is not really malicious, the node might crash for other reasons.

Even then the consensus problem becomes really hard. So it is so if the one of the node crashes, then also consensus problem becomes hard. So this is called a crash fault. So if we have a protocol that works, in spite of crash of some nodes, then we say it is a crash tolerant or crash fault tolerant protocol, but in case it is protocol fails that is it fails to terminate or it terminates but it gives incorrect consequences then we say that the protocol is not tolerant to crash fault.

Now, crash fault is the most benign kind of fault because the node that has that has faulted is not doing anything malicious, just crashed, that now, the one problem in all this in the decentralized setting is that other nodes cannot tell whether the node crashed or he is maliciously keeping quiet to create confusion. So therefore, we have to assume that the crash fault is could be a behavior of a malicious node or it could be really crashed.

A crash fault is the easiest of the maliciousness. The worst kind of maliciousness is called a Byzantine fault. So Byzantine fault basically means that the nodes that are malicious they are not

only going to propose wrong values in the beginning, they are also going to not follow the rules of the protocol. And therefore they will be doing things that are not part of the protocol protocols. What is a protocol is a program for every node.

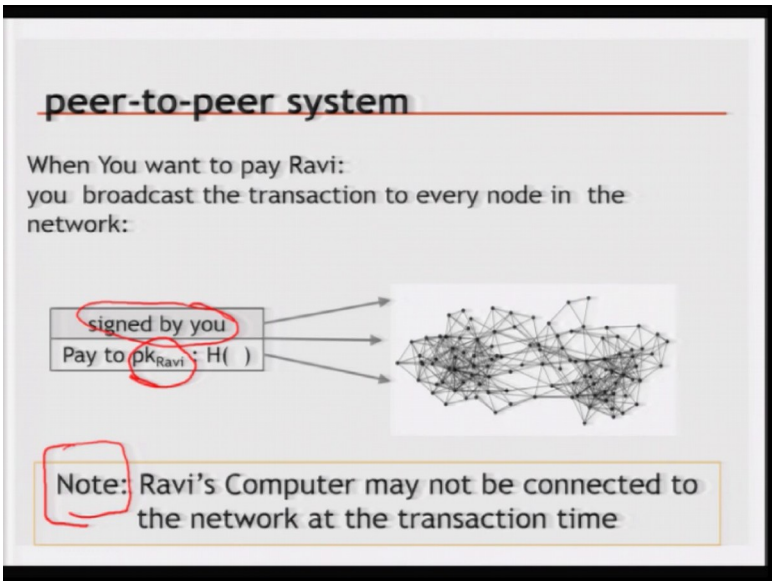
And the, what they and each program here could actually do some computation and communicate it with the other nodes. And then again, do some computation communicate with other nodes. This is what a protocol looks like, right? So a protocol, then obviously has a set of rules or the how the program is supposed to run. The malicious nodes may actually run a different program.

Maybe it is because they have been infected by a malware which has a different program, or it may actually do various things like exchange messages between the other malicious nodes and not ever communicate with others. It can do all kinds of things. So the question is what we want in case of a Bitcoin case for sample we also have to worry about nodes that will work maliciously.

For example, they will validate transactions which are not supposed to be valid or they might actually not valid good transactions, or they might actually create blocks and that those blocks may be having invalid transaction in them and broadcast that block for saying that this is the next block. So they can do all kinds of things. So that in that case also which block has to be eventually added to the network.

Next to the blockchain next is a consensus problem. And the protocol should be robust enough that even if some of the nodes act like that, we have the ability to eventually have a correct block added to the to the blockchain.

(Refer Slide Time: 12:57)



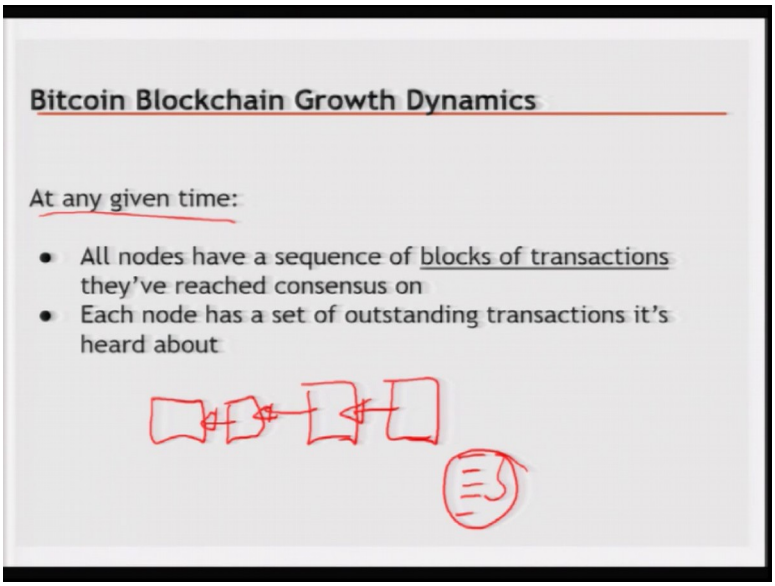
So how the peer to peer system works in blockchain in Bitcoin is that whenever you want to pay to your friend Ravi, some Bitcoin, you broadcast the transaction to every node of the network, saying that this is going to be Ravi's address, which is a hash of Ravi's public key, and you have to sign it. And you have to tell your public key also. And then you broadcast it to the entire network. Now, how the network broadcasted to reach all corners, is a matter of the protocol that is underneath the Bitcoin protocol.

It could be a gossip protocol. It could be other ways of broadcasting it when it can be a multicast and things like that. So now, one thing I want to say is that while you are paying Ravi, it is not a synchronized throw and catch. Ravi may not be connected at that time later on. So all the other nodes will check if your signature is valid. And also it will check whether you have sufficient balance to pay Ravi.

Right the amount that you are telling you are going to pay, all the other nodes will check that and they will validate the transaction. And then eventually some nodes will create a block in which this transaction will be put in. And then eventually, this block will be, hopefully be part of the blockchain. Later on Ravi, when he connects, he will see that there is a transaction which gave him money.

And then after a while, we will talk about that, like how long after Ravi can assume that he can even spend that money. So, that is the way this peer to peer system is working.

(Refer Slide Time: 14:45)

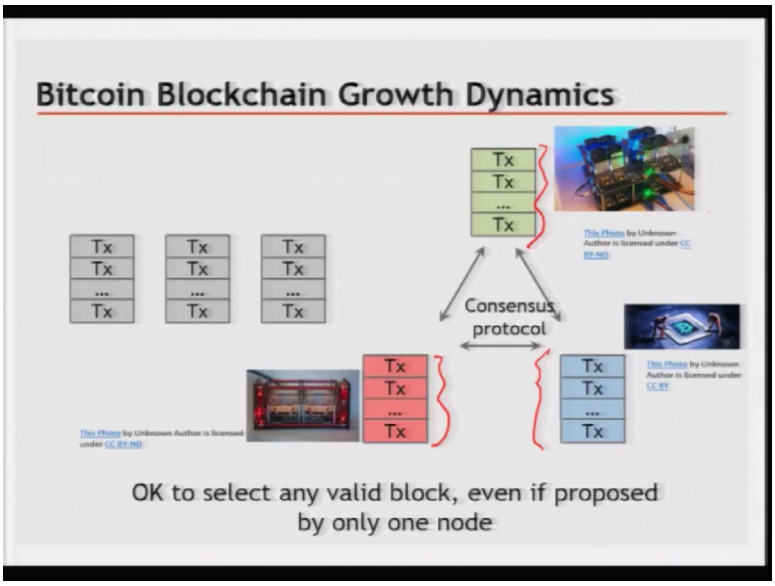


Now, let us look at the growth dynamics. So I have been talking about this for a while, but let us look at it more formally. At any given time, all the nodes that are maintaining a copy or replica of the blockchain will have a sequence of blocks on which the consensus has taken place. And there has been an agreement that these are the blocks that have been made permanent into the blockchain. So, every node will have some set of nodes that are already have gone through the consensus.

And then each node will have a set of transactions that have not been part of any of these blocks. And they have to decide which of these transactions they will want to put in a block. So, they can decide that I will put only this because the blocks have also a fixed size. So, they may not be able to put all the transactions that are still outstanding. So, these are the outstanding transactions, you have to select from them.

And then put in a new block and every node that is part of the process is doing the same thing. They have a sequence of blocks and that are already consented on and then they have a whole set of outstanding transactions.

(Refer Slide Time: 15:58)



When you have this outstanding transition let us say in this example, there are 3 different miners. And they are using, of course, the mining requests a hash puzzle solving, so they have very high computing power, and so on. So green guy decides this set of transactions is going to be in the next block. red guy decides that these are the set of transactions in the next block. And blue guy decides that these are the set of transactions in the next block.

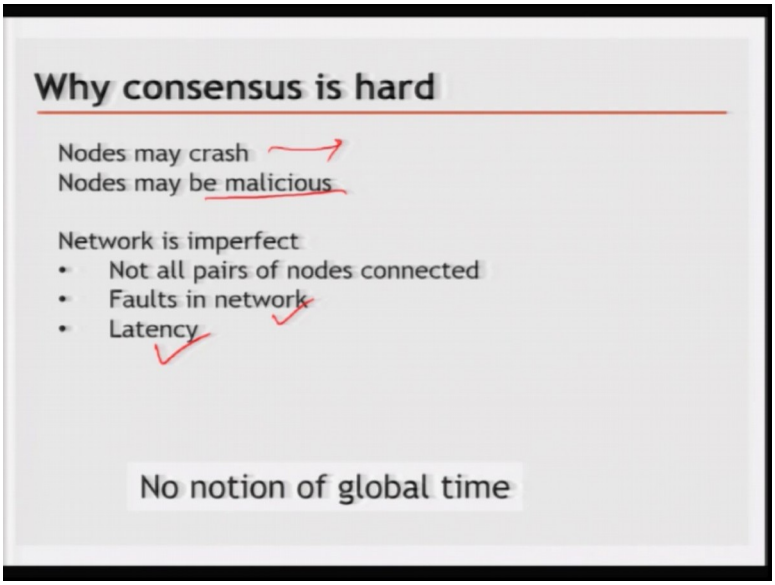
And this there may be many reasons why these 3 are different. One possibility is that the transaction as their broadcast, they take some time to reach all parts of the network. So by the time green guy collected some transactions, some of the transactions have not even reached that node. So therefore, this node, he does not even know the existence of a transaction. This node can also decide that I have this much space, so many kilobytes in a block.

I have too many transactions. I will randomly select some of them. And red guy also has the same issue so he might have heard of this transactions or gotten a copy of the transaction that this guy did not get. But at least he got that so he puts that in but on the other hand for the other transactions, he has randomly chosen from the outstanding transactions that will fit into his block. There may be other reasons, we will talk about that there might be an incentive reason why they choose certain transactions, it may not be fully random.

But in any case, all these blocks will look different. Maybe not maybe some of them will be exactly the same by virtue of probability, but there is no they are not talking to each other deciding which transactions go to the next block. So everybody will propose a block which may be slightly different. Now, when you propose a block, they and if you broadcast that this is my next proposed block, this block should be part of the blockchain.

If everybody does that, there will be a huge amount of data going in and then whoever every node that is keeping a copy of the of the blockchain will be confused because you will get thousands of different block proposals and he would not know which one to take so, there has to be some kind of consensus mechanism through which the next block has to be chosen. So let us see what happens.

(Refer Slide Time: 18:23)



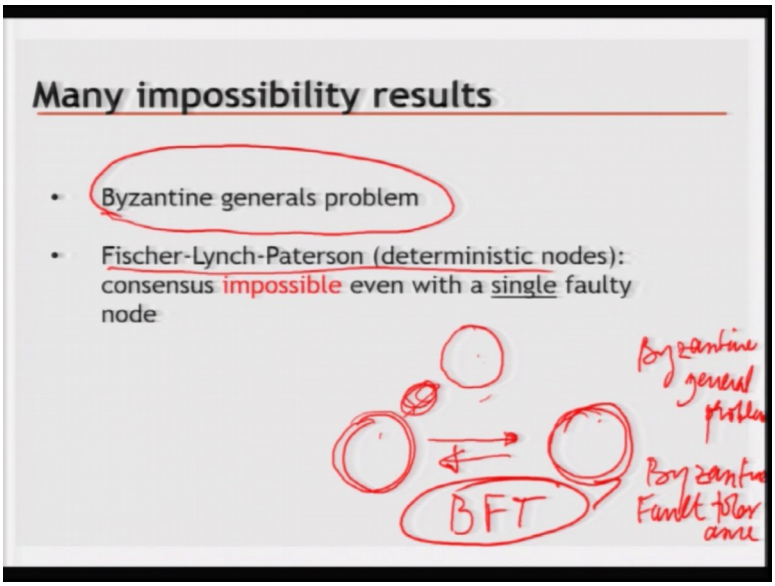
Now, as I said before, that we there may be a crash fault in some nodes while they are participating in the consensus process or some nodes may behave maliciously on top of that, the network has imperfections. So, not all pairs of nodes are directly connected of course, so they there are long paths between them or sometimes there is a segmentation in the network because of various reasons. So, not all transactions are reaching or all block broadcasted reaching all nodes very instantaneously, they might take time.

So there is latency and then the network may have faults because of that maybe routing is taking place and so, on top of that, one could say that well, if everybody broadcast a block, I will look at the time stamp on them. And then I will decide which one has the lowest timestamp and I will take it but that that is also problematic because the notion of time at the various nodes which are geographically distributed is not unique.

So and then the clocks have to are not synchronized, there is no global clock maintained to do this, if there was a global clock, that is every computer has the same clock, then they could have actually done it much more easily by saying that the lowest timestamp one will be taken. But now the timestamps will be not trustworthy because none of the nodes are at different time zones first of all, but even if you adjust for the time zone.

Everybody's let us say, expresses their time as a GMT or UNIX time even then there is a problem because clocks some clocks are slower because of various drifts and skews and other clocks may be faster. So, unless you maintain a global time using GPS clock or some other protocol for time synchronization, this will not work. So, time stamp is not a solution to this problem.

(Refer Slide Time: 20:26)



But while we go and look at the literature on consensus, we found that this problem of consensus is also called the Byzantine generals problem. Later in the course we will actually discuss this in

much more detail. But the idea here is that if you have target enemy site and there are 2 Garrison's which are disconnected from each other, except they can send messengers to each other and they know they cannot win this, the attacking this.

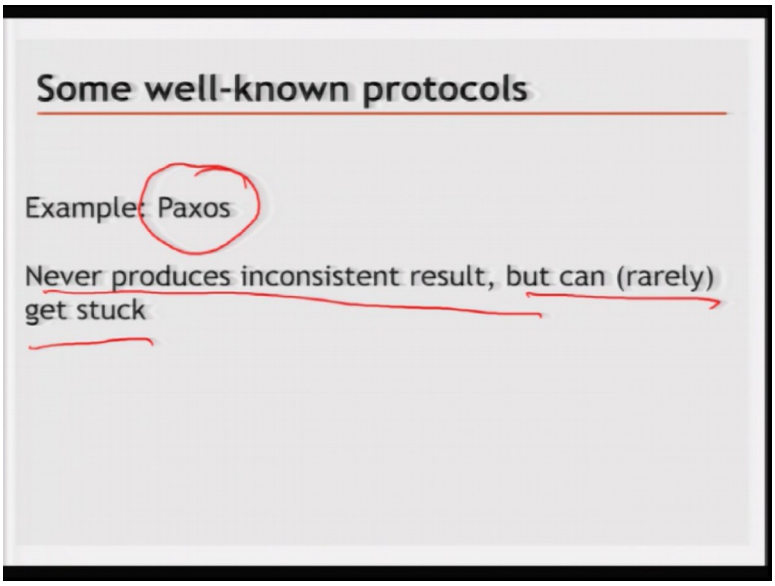
And when this unless both of them actually attack simultaneously from these 2 sides. If one of them attacks and then the other does not, then whoever attacks will be killed completely. So then we have a consensus problem that we have 2 Garrison's with 2 generals. And they have to decide whether they are going to attack or whether they are not going to attack. And either both of them have to not attack or retreat or both of them has to decide to attack.

So this is what is called Byzantine generals problem. Now, the problem here is that if all the generals are honest, and the messengers who were sent and back and forth are honest, then this problem is not a big deal. So, one general decides and then sends to this guy acknowledges and then both are going to attack or both are going to retreat. The problem is that if you assume that one of the general could be malicious or the messengers could be actually changing the messages, then this may actually fail.

So, in that case, they might, this guy might think that the consensus is to attack and this guy is actually think that consensus is to do retreat and therefore, attack alone and then get decimated right. So, therefore, you have to have a consensus that such that the honest generals, opinion should eventually prevail, when the entire thing settles. So therefore, this problem has been a model for this consensus problem as Byzantine fault tolerance problem or called BFT.

So we will see a lot of this discussion and BFT later. So, but early result from 1979, I believe by this is also called FLP result. The FLP basically says that if all nodes run a deterministic algorithm, then consensus is impossible even when a single node can crash. So, this is a very bad news. So, what we are saying is that solving Byzantine generals setting consensus problem, or BFT, Byzantine fault tolerant consensus is impossible even when a single node can be faulty. So, that is an in our case, we cannot even guaranteed single node it could be many nodes that are going to be malicious or faulty so then then what do you do.

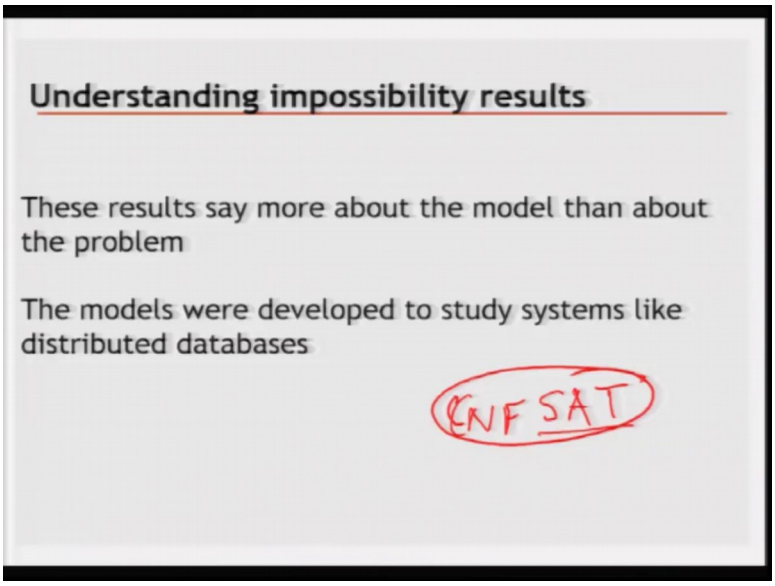
(Refer Slide Time: 23:57)



But there are practical algorithms such as Paxos. So we will also later talk about this. But Paxos is a Byzantine fault tolerant consensus protocol. And we the way Paxos works is that well it never produces an inconsistent result that is the some nodes are saying yes and some nodes are saying no, but sometimes it can get stuck and never terminate. So, remember we said that a consensus problem protocol should always terminate.

And they should always give a result and all the good nodes are non-malicious nodes should arrive at the same result. And that result should be proposed by one of the good nodes. So that was the requirement of a consensus protocol. Now, Paxos does everything else except that it may not terminate sometimes. So you may get stuck but there are other ways to solve this problem.

(Refer Slide Time: 25:02)



So, one thing that is interesting about impossibility results or NP hardness results or any kind of complexity lower bound result is that it is always the worst case scenario for example, those of you know NP hardness so satisfiability sat is CNF SAT is known as NP hard NP complete problem. And it turns out many sat instances are not difficult to solve, but there are very specific types of sat problems which require you know, exponential time.

So, therefore, what, what we can actually say is that impossibility result depends on the very corner case or worst case also, the result of impossibility here in case of this FLP result is more about your assumptions in how the model is supposed to work and what their behavior is, for example, they assume that the model requires that every node does a deterministic activity. So there has been Byzantine fault tolerant algorithms, which work with almost probability one, which are probabilistic or randomized algorithms.

These models were developed to study systems like distributed databases. And now, what we will see as we come back is that this FLP result does not, you know, stop us from actually getting consensus probabilistically in the blockchain So, we will see that the probabilistically the consensus works in blockchain.

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

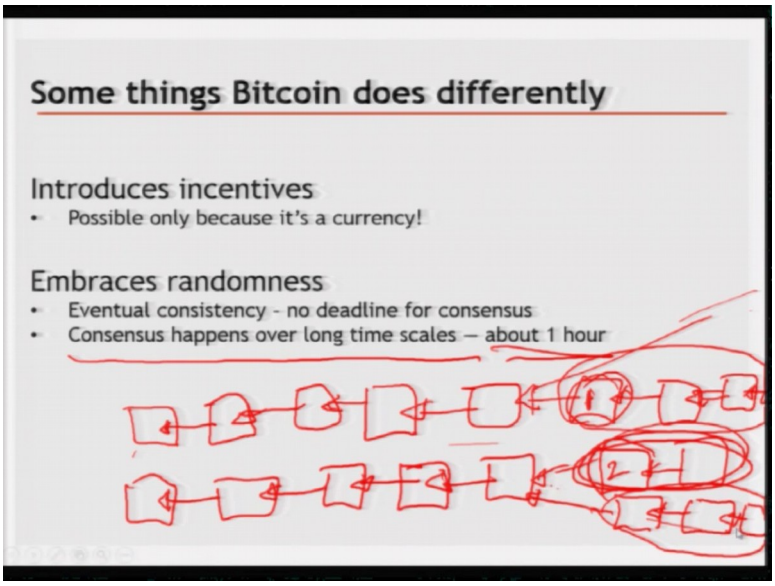
Lecture No. 06
Blockchain Technology and Applications

Welcome back. So we were discussing the Bitcoin consensus problem. And we ended up with discussing the impossibility result FLP. And we said that that is a worst case scenario. And also there are a random that is for deterministic case and there are no match algorithms for Byzantine fault tolerance. But in case of Bitcoin, the consensus works slightly differently than all existing Byzantine fault tolerant algorithms. And actually it works quite well in practice.

And the theory is, not exactly in line with the practice. And but theory is important because you might not; you might think that it is working all the time. So I am fine. But theory gives you the impossibility results are the lower bounds, which can tell you whether you have not thought about some corner case when your system will not work or not proceed, you may not be able to decide on the next block and things like that.

So but we will see now; how this thing works in bitcoin and then you will see how we actually circumvent the problem. So, first of all, as we said before that many times if you change the model of the problem, so in case of a distributed database consensus, for example, which in which context the FLP was thought about, the model does not have a certain elements.

(Refer Slide Time: 01:53)



For example it assumes that all the nodes are either going to do the same thing deterministically. And some of them will be differently, but it does not provide in the model a way to make this malicious nodes to behave correctly by adding incentives. So, in Bitcoin, we add the incentives or rewards for behaving correctly. And if you have that, then your model changes and when your model changes, the existing results may also change because your assumptions have changed.

So, in the original Byzantine fault tolerant problem, there was no incentive. So, we were not saying that if you behave maliciously, then you will not get some reward. But if you did not behave maliciously, we will give you rewards. So, if the if you introduce that, it is a different problem actually, also, instead of, you know, talking about deterministic algorithm you actually embrace randomness in case of bitcoin consensus.

And we will see how that works. But first of all, in the other consensus, the traditional consensus, we assume that this thing will go in through the protocol will go in a round, and then it will eventually terminate. And at that point consensus will be achieved. And then again, another proposal will be made, and another round of consensus protocol will start, and then when it terminates, we will have consensus.

Here we actually relax that when we say that there is no deadline for consensus, we did not say that at this point, the consensus process ended. And this at this point, all the nodes should have

added the same block as the last latest block to their copy of the blockchain. We actually say that eventually when you look at in the long run, be all the block chains replicas should be same, but we are not saying that at every round of consensus at the end, everybody will have the same replica.

So, this is called eventual consistency. And this is another relaxation from the original problem, because we are not saying that, you know, it has to be round by round consensus. So, in fact, in Bitcoin, the consensus happens over a long time scale, in this case, you know, about an hour. But even then about after an hour, the probability of that the different replicas are different becomes extremely low, but it is not deterministic.

And then if you wait for 10 hours, then the probability becomes close to almost negligible. And when we say this, this is how we say this right so, let is say we have this blockchain up to n nodes and another replica also have this. They have the copy of the blockchain. And until then, they have this and until then let is assume that they are identical. Now, another block has been proposed, actually multiple blocks have been proposed.

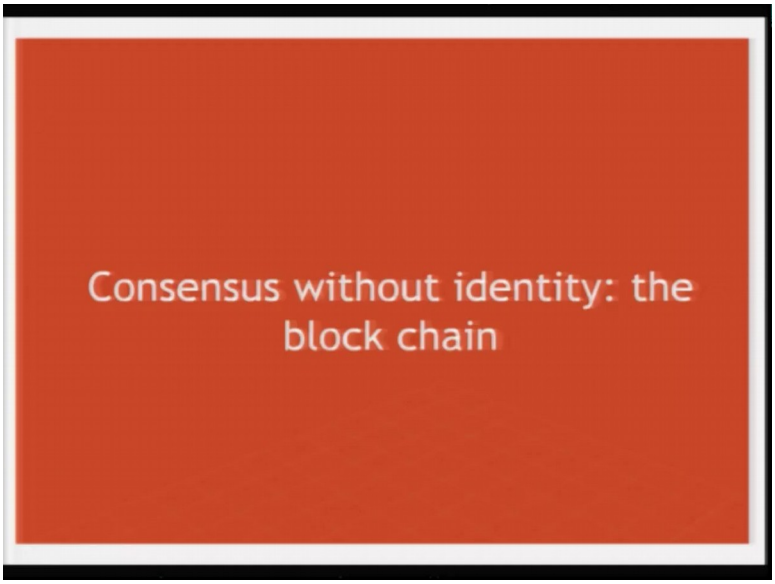
And first guy adds one block. And second guy adds a different block. So 1, block 2 and at this point, I did not have the consensus exactly worked out because they decided on 2 different blocks. But we will see over time, one of these, let's say first guy, after 1 hour will have, let is say, 6 more blocks on top of it, and second guy will have only people, some people built the next block on top of this, but other people said this does not look right.

So, I built I will build on top of this, and then this chain keeps growing, then what we are saying is that at the end, first will become consistent. So, 2nd will become what we call an orphan part of the chain, and the chain will proceed here like this, whereas first copy has always been in the right path. Now, it depending on what 1 and 2 are, it could be that one guy, one guy will grow up and other guy will eventually converge to this this way.

So, at this point, we did not know, but over time over 1 hour when 6 more blocks have been put on top of this one, by then you if it still survives, the one survives and 2 does not have 6 blocks

built on top of this and then there is Alternative Testing, then they will, they will have a one has become permanent. And eventually this guy will have this chain replicated here. So that is the idea of eventual consistency.

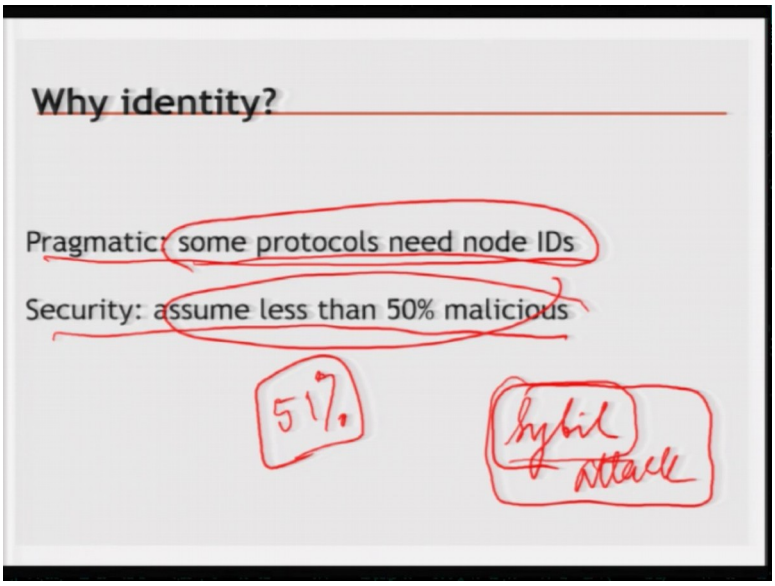
(Refer Slide Time: 07:16)



The other issue is that here we are doing consensus without an identity we did not know who the people are so we did not know, we only know their hash of their public key as their account address. And that is all we know about them. And it may be that multiple people, multiple accounts belong to the same person. So if you are thinking about consensus, but if I have, if I create 10,000 accounts, but I am the owner of all 10,000 accounts, I may actually think that I have more power to overwhelm the consensus.

But especially if I am a malicious node, and I want to overrun the consensus then I can create that many nodes.

(Refer Slide Time: 08:05)



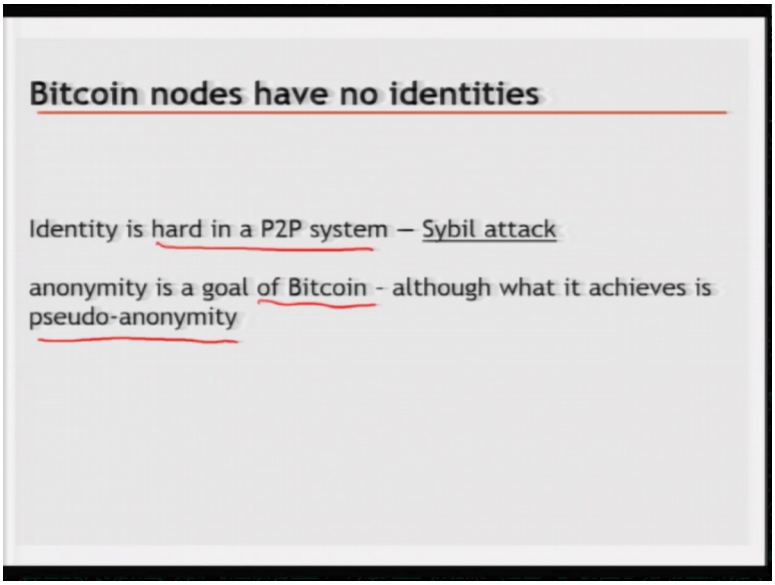
So that when you create multiple aliases for yourself and nobody knows that it is your alias, because there is no way to know and then you get some advantage there is called a Sybil attack. So your protocol should also be robust or resilient to Sybil attack also, right? Because otherwise, you know, somebody who has more malicious intent will create more ideas and then he will he will have more, let is say voting power, for example.

So therefore, we have to actually make sure that the protocol is resilient to Sybil attack. Now, some other protocols require node IDs. And maybe also they connect that to real world identity. For example, we will see when we will look at permission blockchain in permission blockchain every node is actually permissioned. And how are they permissioned? Because there is some way of somebody who actually permitted them.

So there is a central centralization there that who permits them. But after the permission process is done, the rest of the stuff becomes decentralized. But in that case, we know the identity of each node, the real world identity or some other identity, in this case in bitcoin and ethereum is called permission less, and we did not know the real identity, and somebody may have multiple identities. In fact, we saw yesterday that the change address, for example, requires that you create your own multiple identities.

So, under these circumstances, we have to make some assumptions. So we assume that we already talked about this that we have to always assume how many of them are malicious, we cannot say 100% could be malicious because then the protocol would not work. So we assume that less than 50% are malicious. So in that assumption, under which the Bitcoin protocol works, if anybody gets 51% malicious, then and these malicious nodes actually collude with each other, then the protocol would not work.

(Refer Slide Time: 10:15)



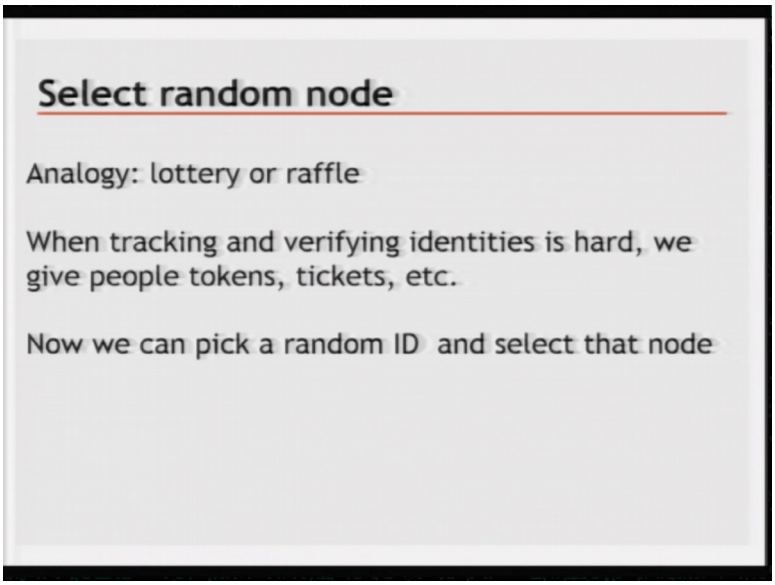
So, as I said that in a P2P system permission free permission less so, we cannot really determine who and when we are making the P2P system in such a way that the there is a low barrier to entry. So, anybody can create a public private key pair and enter the system. In that case, Sybil attack may happen. On the other hand, the goal of Bitcoin creation was also not only decentralization, but also give anonymity.

Now, in the very beginning that this anonymity is not real anonymity. It is actually pseudo anonymity because people are using various data mining techniques because all the data of a blockchain Bitcoin blockchain are publicly available. So, you can actually look at the correlation between different addresses, how they transact with each other, and so on. And they can actually determine that these addresses seem to be from the same entity and, things like that.

So it is in a sense that it is pseudo anonymous, but, as I said that it is almost impossible to establish the real identity even if you do all this to know which accounts are together and which accounts are not together, but you cannot really still know who the real person behind that identity is so but that is a goal of the bitcoin. Now, we saw that this goal also has some downsides, which is the use of bitcoin addresses as ransomware addresses.

The use of bitcoin in the drug trade and another dark wave activity, so that is a different issue here by anonymity is the reason why identity is not there, which makes the consensus even more challenging.

(Refer Slide Time: 12:05)



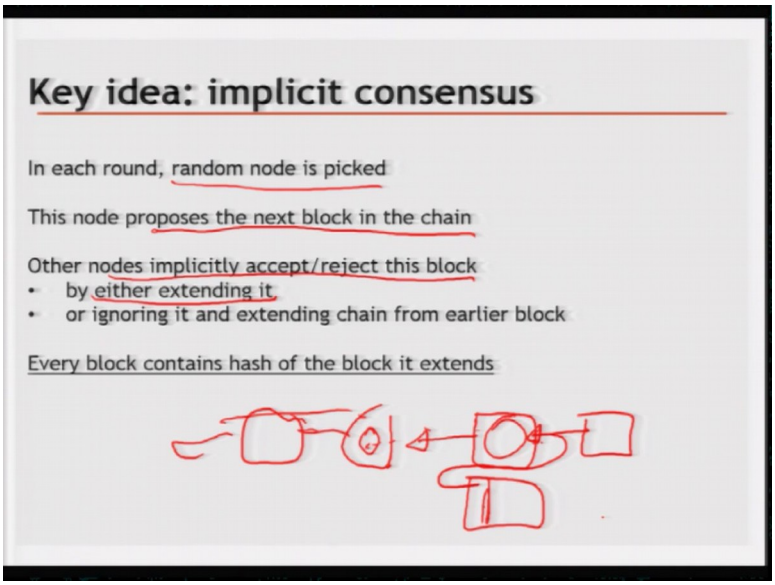
So the way the consensus works, mimics the following process. Suppose there is a raffle like you go to a concert. And then at the end of the concert, there is a raffle. And they will give away a couple of prizes by picking your name from some kind of hat or some box. Now, they did not know your names, they did not know your identity. So what they do is that they look to take at the stub of the ticket that you bought, so it has the same ticket number.

And then the put the ticket number in the box, these stubs of the ticket in a in a box and pick one randomly, and then say, whoever holds this wins. So in there, your real identity is not required. You are given randomly, some number, random ID, and then I am picking one of them. So suppose you want to decide that way I will do the consensus is that everybody proposes their

blocks. And then I will give everybody a random ID. And then I will pick one of the random ID, and then say that your block will be the next block.

So this is what we want to do but, again, we did not have a central authority to do this if there was a central authority, then the central authority can arbitrarily decide whose block will be taken central authority could have done a raffle kind of thing to pick them, but central authority is not there. So we have to actually do it in a decentralized fashion, we would want to do this, but we want to do this in a decentralized fashion.

(Refer Slide Time: 13:41)

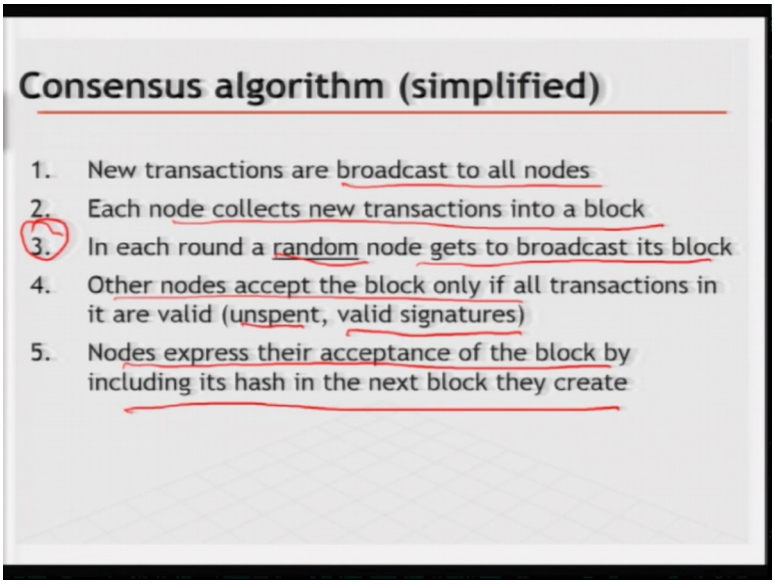


The key idea is that in each round, a random node is to be picked. And this node will be allowed to do give the next block to the chain. And other nodes will then implicitly accept or reject this block by either extending it so when you put a new block to the existing chain. The others may decide that they will extend your block or they might say, I did not think that this block should be there. So I will go to the previous block and extend from that. And by extend, I mean that when I create a new block, I put the hash of one of the previous blocks.

So I can put the hash of one guy, or I can put the hash of other guy and then conceptually, if I do the hash of latest guy here, then I am connected to this and basically I am extending this part of the chain from here. And if I connect to the other guy, then I am extending the chain from here. So that is how I say all the other nodes will say that I accept you as the latest block. So every

block decides every new block that is created will either extend the latest new block or it will extend the previous block and that accordingly the chain will grow. So this is how the system grows.

(Refer Slide Time: 15:04)



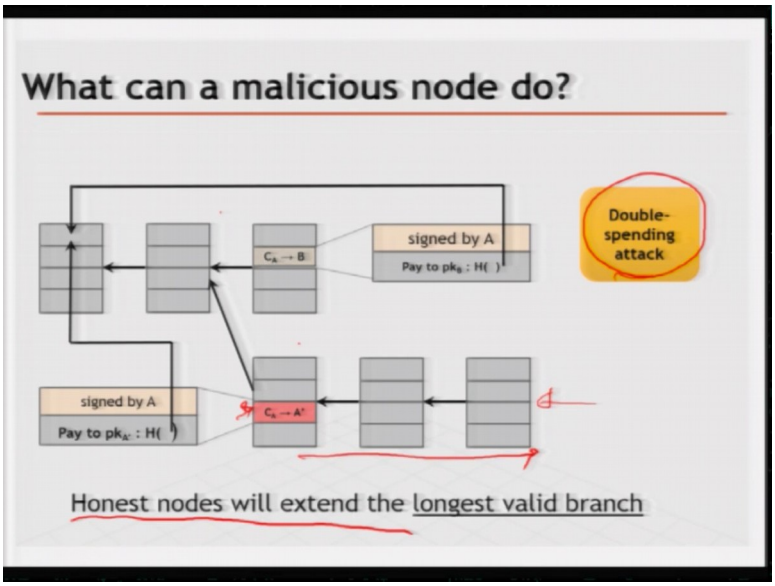
So a simplified form of how this works is that all new transactions are broadcast to all the nodes through the peer to peer network. Each node collects new transactions and decides to create a block. And as we discussed that every node may have collect a different set of transactions into a block, and the blocks are fixed size. And in each round, a random node gets to broadcast its block. Now here we are, we are not saying the whole story because which random note gets to broadcast is blocked is something that we will leave it for later.

All the other nodes accept the block only if all its transactions are valid. Now if you have sent a block in which there are invalid transactions, then no node will accept it except for maybe malicious nodes, which is that money that you are stealing has not been spent before that is you are not doing double spending. And they have to check whether the signature is valid, then they have and this is they do for all transactions in the block.

And they say that, this block is valid. Now, the nodes then will express their acceptance of the block by including its hash in the next block they create. So, that is the overall scheme of things.

Now this step is what we have to discuss more because how is this random node gets selected, because we have no central authority, so we have a decentralized system.

(Refer Slide Time: 16:39)



So before that, let is see what a malicious node can do. So let is say I have the blockchain until this and then I make a transaction which makes into the next block. So I am giving some money to be and this is CA is the coin that I have right now. And I sign it so it is a valid transaction. Now, what you can do is that you can try a double spend so, you say, give this money to my other address. So, A might decide that. So B let is say a merchant.

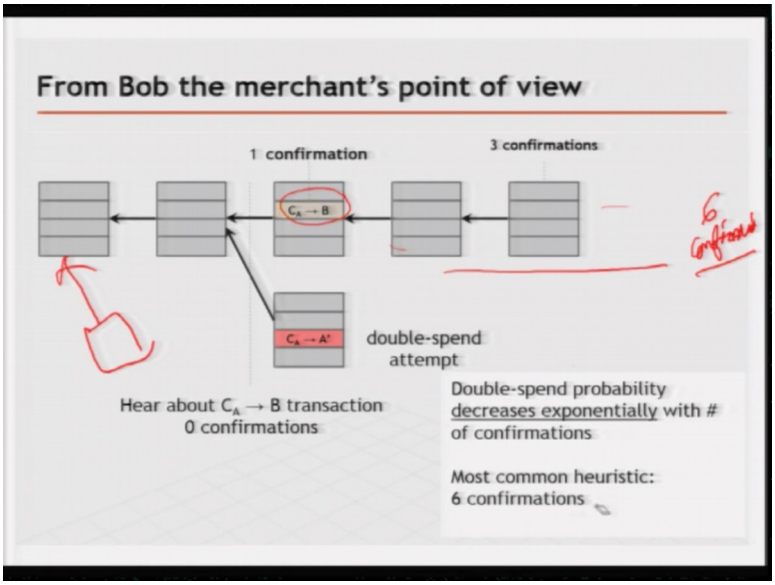
And you say to be that send me this package, I am going to buy and I am going to pay you in Bitcoin. And the way you know that I have paid in Bitcoin, you listen to the broadcast, and very soon you will see the transaction where I am saying that I am paying you. And then also, if you wait a little more, you will see that in the next block, if the merchant believes after seeing the next block that he has been paid, and ships the merchandise.

At this point, what A can do is that once he gets acknowledgment that the merchandise has been shipped, he can actually then put another transaction and then get, which pays the same money to himself in another alias. And then make sure that all his friends and others are built the other blocks from the block that contains this second transaction. So then first transaction, first block becomes an orphan block, if somehow this block becomes extended.

And that is a double spending attack. So an attacker, in this case A might actually then in the meantime, the merchandise has already been shipped. So the merchant after a while the merchant goes and checks the block blockchain again and sees that this has become orphaned in the current latest blockchain does not contain that transaction. And therefore he cannot spend the money he got because his transaction is not there in the actual blockchain.

So these doubles spend and attack is a big problem. So an honest node is supposed to extend the longest valid branch. So, if this becomes a valid branch see if this double spend has can be considered a valid branch because other branch has become orphan. So, this is the honest nodes may start building from here and therefore depriving B of his payment.

(Refer Slide Time: 19:20)



So, the merchant's point of view this is really a loss because he is not getting any confirmation. So, what we say is when the block mix when somebody makes the transaction part of the block and that block becomes part of the block chain, and then we say it is a single confirmation. Then when somebody builds another block on top of this block, then we say we get another confirmation and then we get another block built on the same chain then we say the third confirm confirmation.

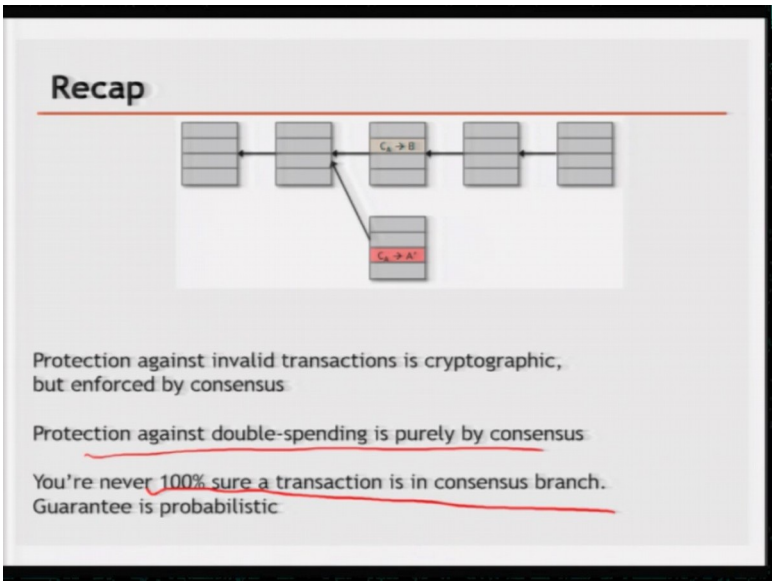
So, if the confirmations keep coming on the original one, then Bob will be okay because after enough number of confirmations, other block will become orphan and so therefore this attempt to double spend will be getting no confirmation. But in case there is their confirmations keep building. So normally in Bitcoin until 6 confirmations, you did not ship your merchandise. So that is and we will see later that it takes about 10 minutes for one new block to be mined.

So, 6 confirmations will take 60 minutes. So that is why we say that you have to wait until 1 hour see that the transaction that you are interested in, the block which has 6 confirmations, then you might more or less sure that people cannot really create a side chain, which will grow longer because then another 1 hour has to pass before this can be done, but honest nodes will not build because this is a smaller chain. So they will start building on top of this.

And therefore this chain will keep growing. So this is all probabilistic because you are assuming the honest nodes listen to, they actually abide by the protocols. You are also assuming that nobody equates later or 51% power, or 51% nodes did not become malicious and subvert the chain by changing from here itself, like they can start building here, all kinds of stuff can happen, but the probability is very, low.

So eventually, even after 6 confirmations, there is a very, trivial probability of being subverted. But we live with that and normally it works. So, double spend probability decreases exponentially with the number of confirmations, and this can be proven mathematically, because the process of new block creation usually follows a poision distribution and therefore inter block arrival time is an exponential distribution. And so most commonly heuristic is to wait for 6 confirmation so which is closely an hour.

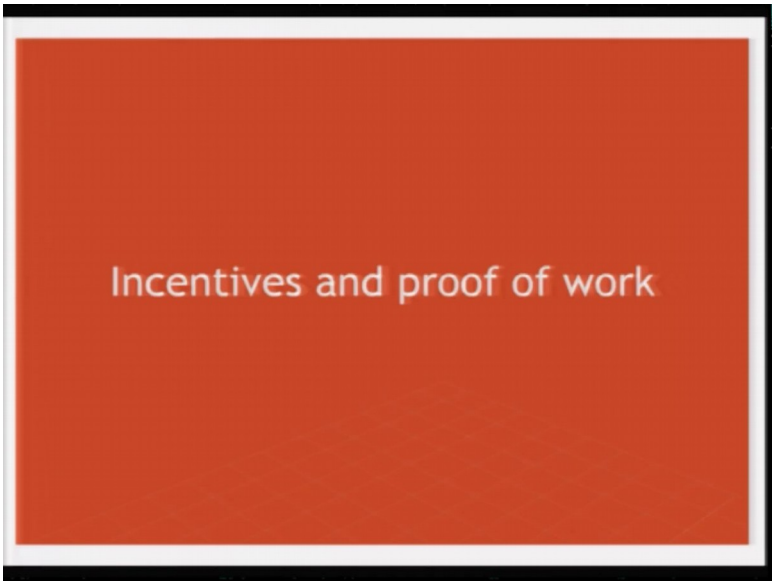
(Refer Slide Time: 22:04)



So to recap, the protection against invalid transaction is completely cryptographic, enforced by the consensus. Protection again, double spending is purely by consensus. So when you check for transactions validity, the consensus does not come into play. If a transaction is invalid, every node will see that by checking the signature and whether the money belongs to you and things like that. And that can be done all with cryptographic techniques.

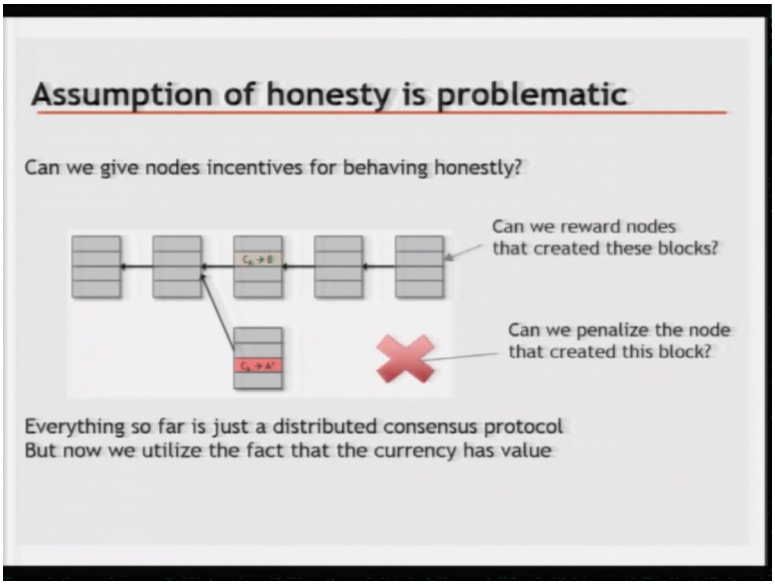
Protection against double spending is actually has to be done through consensus and it is probabilistic. And you are never 100% sure that your transaction is in the consensus branch. The guarantee is probabilistic, but with overwhelming probability.

(Refer Slide Time: 22:51)



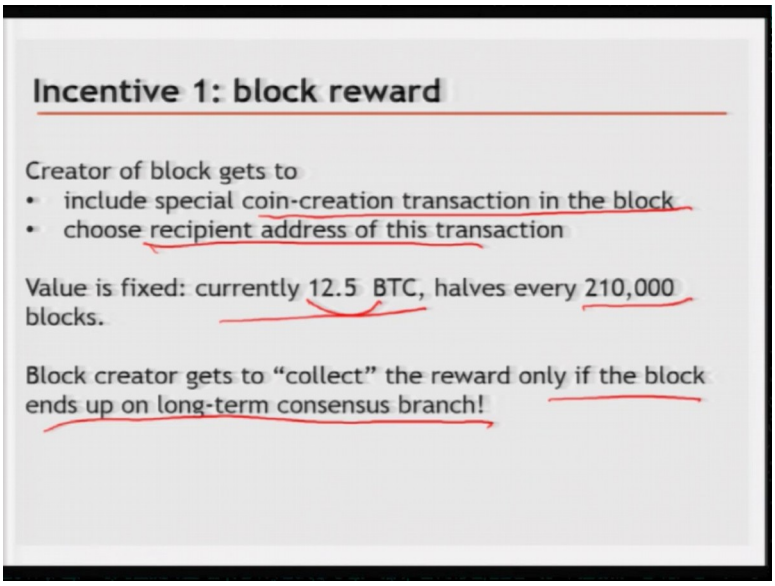
So that is what we have seen so far. And now we talk about the incentives. So incentives are actually important part of the problem of consensus in blockchain. So we talked about this earlier, that unlike the classic system of consensus, distributed computing, Byzantine fault tolerant computing, there, we did not try to change the malicious nodes behavior towards behaving correctly. But in here, we actually do that by giving incentives. So let is look at the incentive mechanism that was put in place for this.

(Refer Slide Time: 23:32)



So can we give nodes incentives for behaving or behaving honestly, or can we penalize the nodes that behave dishonestly? Now, if there was a centralized system, somebody would be actually determining, this behavior is not acceptable, this behavior is acceptable. So we should penalize, but there is no central agent or central authority to do that, and therefore we cannot do this directly. So we have to have some mechanism for incentive. And those who are not behaving correctly will not get the incentive. So that is how it works.

(Refer Slide Time: 24:14)



So therefore, what we were going to do is that we are going to show you 2 incentives. One is that when your block gets added, you can add a special coin creation transaction in the block and put the recipient address yourself. So whoever creates a block, he assumes that his block will be accepted. So in that block, he adds an extra transaction, which we call a coin creation transaction, which creates new coins. And then he say puts the recipient addresses one of his addresses.

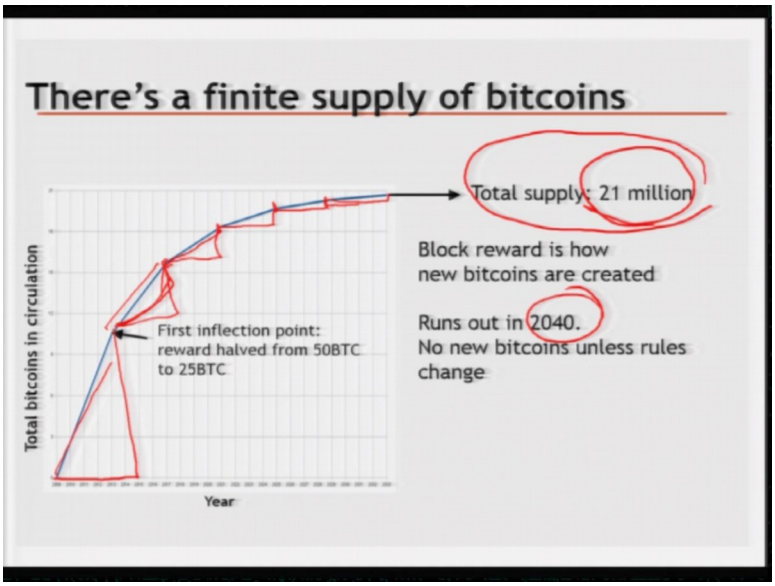
He has many aliases, he will put one of these addresses. Now how much he can create, but the coin creation transaction is fixed. Right now it is a 12.5 bitcoins, but it has every 210,000 blocks. And so far with the 10 minute in interval between blocks, it is roughly about, I think 4 years. So therefore, right now if I create a block, I will put a transaction saying that create 12.5 new bitcoins and give it to my address.

Now; if I, my block gets accepted, then that transaction becomes part of the blockchain. And then after 6 more blocks are built on that block I get to use that 12.5 blockchain if my block does not get added to the to the blockchain, or if it gets added, but nobody builds on top of it and I did not have 6 confirmations, then I cannot use that. Those block chains that I created even though I put it in the block.

So the block creator gets to collect the reward only if the block ends up on long term consensus branch the branch that has many, others built on top of it. So, at least 6 blocks has to be built on

top of that, in order for this to be able to use that 12.5 Bitcoin earlier it used to be 25 Bitcoin and in the beginning it was 50 Bitcoin and later on it will be 6.75 Bitcoin and so on.

(Refer Slide Time: 26:19)



So, and there is a finite supply of bitcoins. So, the only way to create Bitcoin is by this process your block has to be accepted, you will get reward and this way every 10 minutes somebody or the other will get some rewards and then more coins will be created. Now, in the beginning, as I said it used to be 50 per transaction, so the slope was higher of the total number of Bitcoin in the system. Then it became 25.

So the slope was lower, and then now it is 12.5 the slope was even lower, but you see the number of the time required to reach that 210,000 blocks is shrinking. So, this is 210,000 blocks were created in this more time here it is much smaller and this to 210,000 blocks require a lot longer I think. So, therefore, these things and then it will become 3.7 to 5 or something and, so on. So, eventually it will taper out 2040 I then the total number of bitcoins will become 21 million.

And beyond that no more bitcoins will be created unless the rules change. So, this provides the Bitcoin ecosystem some scarcity right because if you have a finite supply of bitcoins, then the prices will not be high right the exchange rates will not be high. So, therefore, in order to build in this scarcity, they have decided in the beginning that there will be only 21 million bitcoins

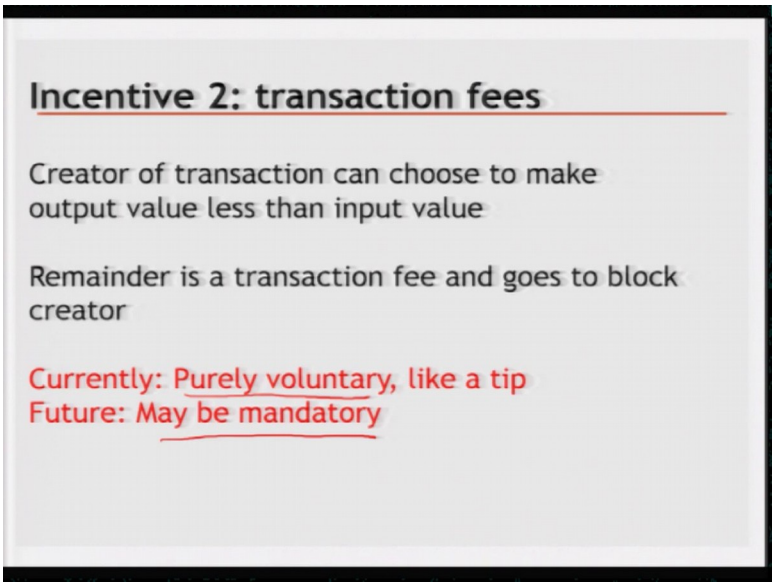
created. So, we will talk about the other incentives in the next session. At this point we have learned how to do one type of incentive for, nodes who play by the protocols, rules.

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

Lecture No. 07
Blockchain Technology and Applications

Welcome back to this session where we would be continuing with the incentive mechanism that is used in Bitcoin blockchain. So last time, we spoke about using incentives for making the participants or the miners behave according to rules by allowing them to earn some incentive.

(Refer Slide Time: 00:34)



And we also talked about first type of incentive which is the block reward. So if your block gets selected, then you can pay yourself some very fresh Bitcoin. And now at this moment, this number of coins that you can win is 12.5 bitcoins, but it started with 50 bitcoins and then it started it became 25 bitcoins, and now it is 12.5 bitcoins so that is and bitcoin price as of now, as of today is about close to $8,000. So, therefore 12.5 bitcoins is quite a bit of money.

The second type of incentive is transaction fees. So, if you want your transaction to be put in a blockchain then you want to have you want to pay some transaction fee and more transaction fee you pay it better incentivities for a miner to include your transaction in the block. So, what you

have to do to pay transaction is that you have to remember in bitcoin you basically say what are your inputs and what which input which transactions you are inputting from.

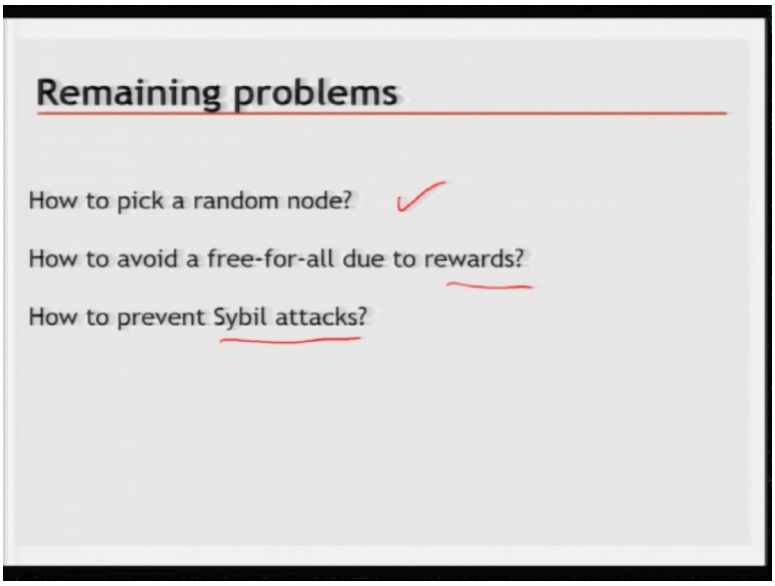
And those inputs from the, which should be from the output of that previous transaction and then you have you specify output where the money is going. So and the rule is that the total input, a total number of coins in the input should be equal to total number of coins in the output. That is how you create a change address and you if there is any left, you put it to that you add directed to that address which is under your control.

Now, if you want to pay transaction fee, what you have to do is that you make the output value that is all the different addresses where you are paying from this through this transaction should be less than the input that you are putting in. And therefore, the rest of the amount that you are not saying anything about can be collected by the miner. So every miner who creates a block will calculate from each transaction how much transaction fee it is getting.

And then it will put another transaction that gives that money to an address that is controlled by the miner. Now, not all miners will be winning. So, whoever wins that round whose block is made into the next block will actually be able to redeem that amount? So, right now, it is purely voluntary like a tip, but in the future it may become mandatory because as we said before, there will be only 21 million bitcoins and by 2040 the rewards will stop you know will become zero.

Basically, because we are reducing the reward number of coins that we reward, you know, by half every 210,000 blocks and therefore, at that point the transaction fee should be the only incentive for somebody to include a particular transaction into the block that they mined.

(Refer Slide Time: 03:56)



So, now we have seen the problem of malicious or rogue users or rogue miners and we have seen that incentives is one way of engineering their behavior. Therefore, they might actually behave correctly because if they did not behave correctly, they have no chance of winning the block and therefore, that they will not get any of this rewards. Now, there are still more problems to be solved in the consensus.

First of all, we said that every round, every time a new block has to be added, everybody is creating their own block and each of these blocks could be different from each other. So we said that, we will actually use a raffle kind of mechanism everybody will be given some kind of a random ID and one of the ID will be chosen as a winner and that winner's block will be propagated and added to every copy of the chain.

Now, it is not as easy in a decentralized system as you might think. Because there is no central authority to assign random numbers, random IDs, and then there is no central authority to actually carry out the selection. So it has to be self-selection kind of process. The other issue is that when a block is when you choose somebody's block then that even if there was a central authority to choose that, that minor.

Even then the propagating that that information throughout the entire network, which is vast, would actually take time. In the meantime, some other person might actually consider himself or

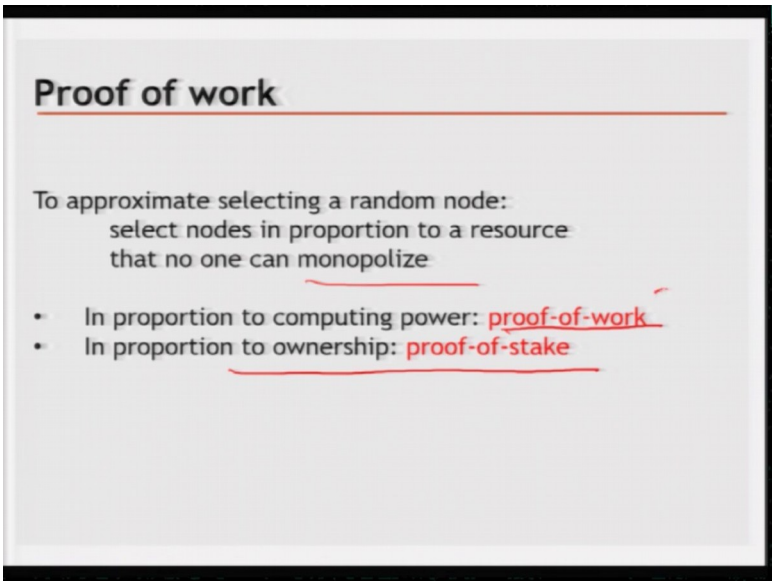
herself as a winner, and then start adding the block in its vicinity, and therefore, there may be a race condition. That is so, we will see how that problem is solved. Second is that you know, if everybody tries to mine, then there will be a huge.

You know, number of races and how to resolve those races in order to avoid such a situation. And then there is this issue of the Sybil attack that if somebody creates many, IDs, then he might feel that his probability of being selected this way would be more. So, therefore, we have to do some mechanism by which having more ideas does not help you to have a higher probability of winning the race.

Let us see how the random were node is selected. So, and this mechanism in bitcoin particularly or in also in ethereum is proof of work. So what you do is that all these different nodes have different resources at their disposal. So computational resources, memory, the GPUs and specialized E6 hardware, specialized FPGA and all that stuff, and then they accordingly they have invested and also as I said before that to be keeping that interest.

You know large scale computing infrastructure running, they have to pay for electricity they have to pay for cooling. So, you have to kind of decide who wins based on the random node will be selected based on the proportion of the computing power that a node has so, therefore, if a node has 30% of the computing power, then he should have a 30% chance of being selected as the winner.

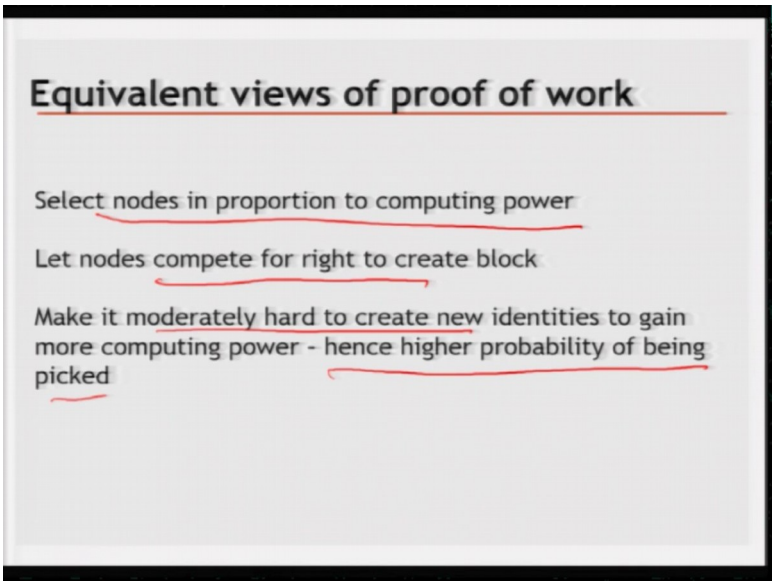
(Refer Slide Time: 07:55)



So, let us see, and this is called how do you know the how much resources they have so, you have to ask them to solve some puzzle. And that is where the hash puzzle comes in. And then whoever does it, whoever has more computing power will have will do it faster. And therefore time taken to solve the puzzle kind of indicates how much computational resources they are putting in and therefore, it is called a proof of work.

So proof of work is proving how much computational resources they have there is also other possibilities, for example, ethereum now also has proof of stake, proof of stake is in proportion to ownership. So how much you already have acquired that according to that your proportion of being selected is decided. But in Bitcoin, it is all only proof of work. So we will know now focus on the proof of work.

(Refer Slide Time: 08:45)

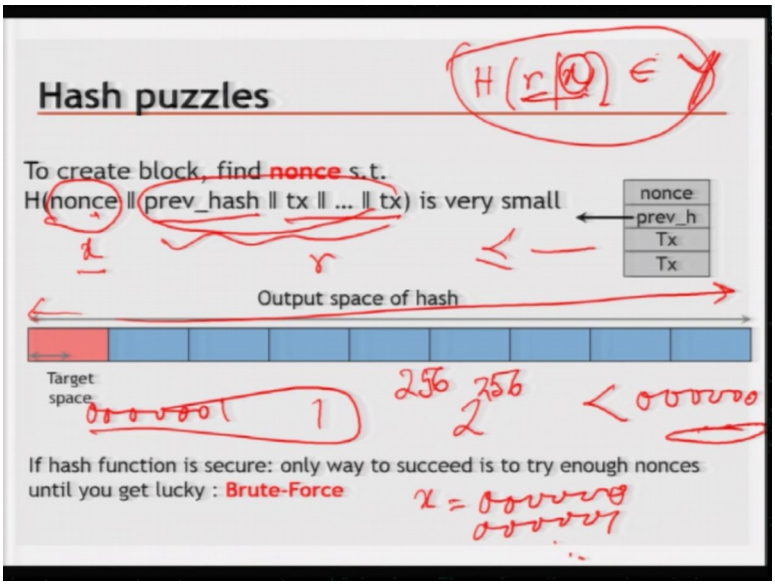


So what is happening with proof of work? So you can, as I said that proof of work basically allows you to select nodes according to the proportion of computing power they have for the inside the entire infrastructure. So to prove that they have that computing power they have to compete for right to create the block and it should be moderately hard to create new identities to gain more computing power. So, hence higher probability of being picked.

So, by creating more identities will not increase your computing power because you have a fixed amount of computational resources. So, you have you can create 1000s of ID's but then that computational resources will be used by all of them and therefore, total computational resources you have will be the same so, that basically, focuses us on this on avoiding the Sybil attack. So, so, if you have invested so much that you own the 30% of the total computational resources in the entire network.

Then dividing yourself into 300 different identities will not change that from the 30% to you know, 300% or something. So therefore, you have no way to increase your computational power. And in that sense your probability of being picked as the next block. So that is, how it is done. So let us see how it is really done.

(Refer Slide Time: 10:19)



So, we talked about hash puzzles, so remember hash puzzle was, we said that if you have a random number, and you challenge them to give me give you an x, so that that random number concatenated with x will have the number will belong to a set y. So y is the target set. And then you have to find an x given R, what is the, hash such that it belongs to y. So that is what it is what is done here. So, you take the block, so you take the previous hash and all the transactions in that block, and then you add x.

So this is your x. And this is your r basically in the hash puzzle set, formal setup. So you have to find an x such that with this r, you will be able to do

$$\mathbf{h(r \mid x) <= y}.$$

And that means that is a set of possible hash values. So if this is the output space of the hash values, let us say you are using 256 bit hash values, then your output spaces $\mathbf{2^{256}}$. So that is very large this from here to here.

But you say that, if you will be winning, if you are the when you come when you choose an X such that, that your hash value is basically less than, let us say, so many leading zeros followed by any numbers. So which means; that you are saying that not any number from $\mathbf{2^{256}}$ possible

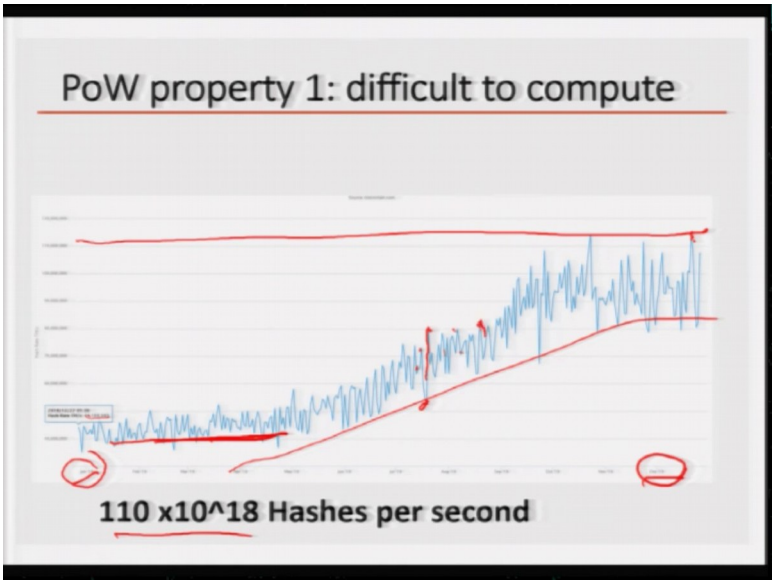
numbers can be selected. Only the ones with so many leading zeros, which mean relatively smaller numbers, any x that will satisfy this will be winning for the hash puzzle.

Now, many miners let us say there are 3 miners each having 30% of the resources. So there is a probability that all three will solve them almost at the same time. And then when they solve it, they think that they have won. Because, there is no central authority to say that you are the winner. So, all three will have their blocks as competing blocks for being added to the blockchain. So therefore, there will be a race condition.

But before we go into the race condition, you see that the hash puzzle has this property that we discussed earlier, is that the only way to solve this is through brute force. You start by saying that, x = 0000 will be this. And then when you try to hash if the hash does not come out, to be in that range, then you try 000001 and so on. And then you have to, you may have to try all of them. And depending on what you have chosen here, as your list of transactions that you are putting in the block.

You might actually have an issue with respect to there; you may not even get there even if you exhaust all the possible things. So, but by the time you exhaust all possible things; somebody will hit on the right x for his choice of the block, and then he will be one of the potential winners.

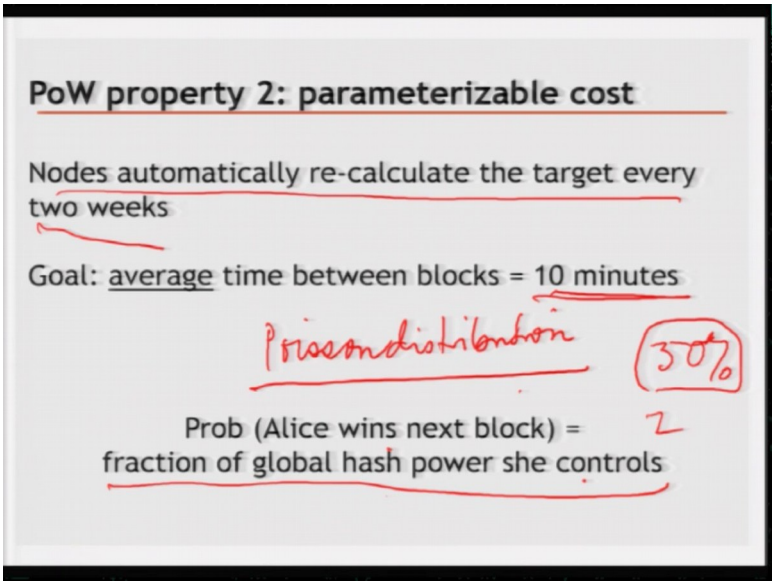
(Refer Slide Time: 13:31)



So, now everybody's competing hash so, the every second now, I looked this up and in December 2019, as you can see, that over almost like **$110*10^{12}$** hashes per second. So, this is Tera hashes. So, that many hashes are computed per second by the blockchain ecosystem. Now, all these hashes are not computed by single node all nodes are trying. So, this number of hashes per second is actually the total cumulative number of hashes computed.

And some of them will be doing much less number of fascists and win. So, but the point is point here is that the proof of work is a very difficult computation brute force computation and there is a lot of computation that goes on at every mining node that is trying to solve this hash puzzle.

(Refer Slide Time: 14:30)



Now, the problem is that in the beginning, everybody was using desktop computers and then the hash rate was much less as you can see even within like this is from January 19 to December 19. In January 19, the hash rate was 40 million, tera hashes and so if you look at this, is 44 million. Here we are looking at here we are looking at almost 110 million. So, within a span of a year the hash rate has increased.

Which means that people are throwing in a lot more resources, a lot more parallel computation, a lot more GPUs and so on. To do this hash computations and therefore, we are seeing a surge in the hash rate of the entire network. So, therefore, what will certainly happen is that in the

beginning, the let us say I keep I give you a hash puzzle, and you can solve on an average within 10 minutes, then after you throw in more computational power.

You can do try parallely more many more hash combinations, many more nonce combinations, and therefore, you will be solving it faster. And then more computation you give more computational resources you give you can compute even more, you know, efficiently. So therefore, what the, what it happens is that every 2 weeks the nodes automatically recalculate the targets set. So how many leading zeros you are going to require to have in your hash to when the hash puzzle keeps increasing.

So if you see here, let us say this is the target space today, they after 2 weeks, we find that the number of hashes per second has increased. So it will be solved faster. So we will decrease the space a little bit and this will be the new space and then it will become new space. So, you make it harder and harder to reach this part of the hash output space through your computation. Now, the way it is done is that we want the average time to actually mined a block should be roughly 10 minutes.

So usually this follows a poision distribution. So, arrival of new blocks follows poision distribution. So therefore the time interval between blocks follows an exponential distribution. And this exponential distribution, we want it's mean to be 10 minutes. So that this adjustment that is done automatically. Now this adjustment is roughly in two weeks. So if you if for multiple weeks, you see that the hash rate is not going up, then you did not need to adjust it.

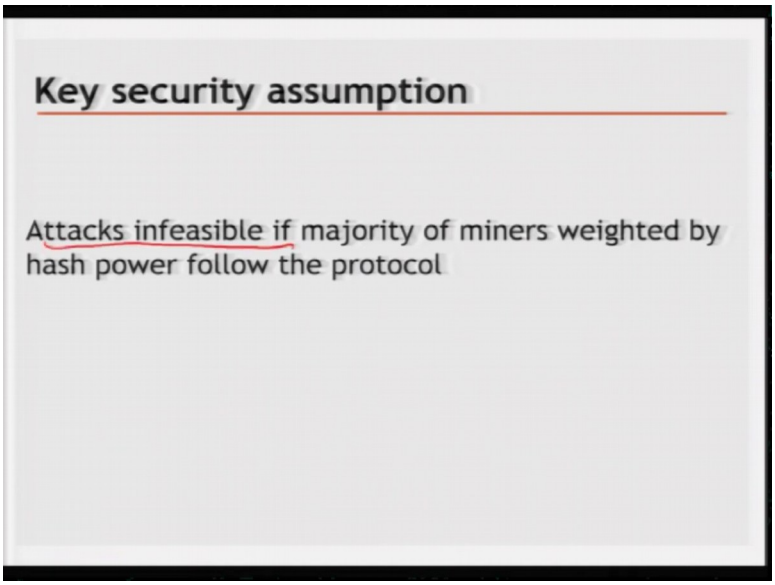
So you actually can see that here there was a month from March, April, the hash rate was pretty steady. Actually, it was pretty steady here. So you did not necessarily have to adjust it. However, as soon as you adjusted the time, the hash rate goes down, so this is the hash rate. So these are the points when you did adjustments. So you see this is a repeated pattern. So these are the points when hash rate went down because you made the problem harder.

But then they learn, you know how to get it faster. And then over time, the hash rate increases, then you again adjust and your hash rate goes down. So this is how the thing is dynamically

adjusted. So probability that somebody wins, the next block is the fraction of the global hash power she controls. So that is how it is designed. So if you throw in more resources, your probability is going to be higher.

So if you have 50% of the resources, then you have a chance that every 2 blocks you will be mining one block, so that is actually quite on the edge. Because at that point, you know, if you have more than 50%, then you might actually be doing more blocks than anybody else. And therefore, you can actually bypass the original chain and create more blocks and build another chain which is longer and that is the 51% issue that we talked about.

(Refer Slide Time: 18:49)

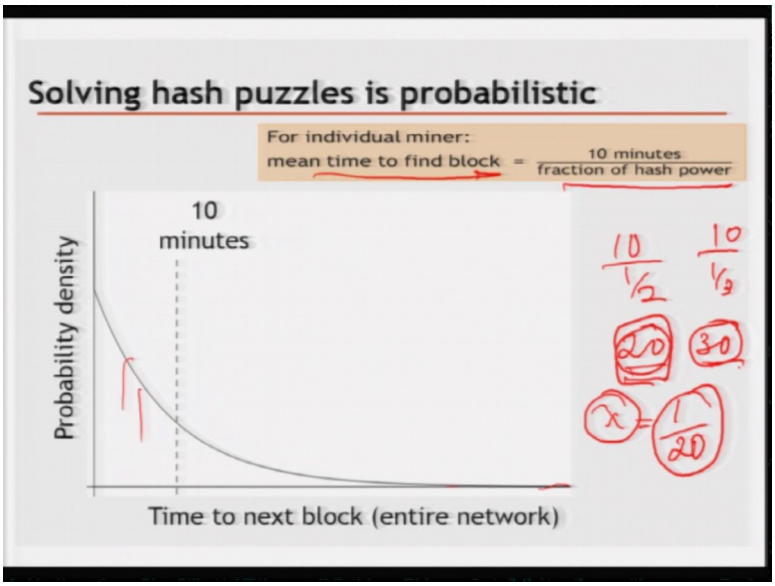


So therefore they attacks or 51% attack is not possible. If majority of the miners as weighted by hash power, follow the protocol so it is the number not the number of miners that we are worried about, we are worried about the total number of hash powers they have. So if somebody has less than one person hash power, there may be, you know, you will need 51 of them to get 51%. If some, people have like 10%, and then you need about 5 to 6 of them to get 50% hash power.

So, it is not the number of miners, but how much hash power or how much computational power they have together, if they collude to become 51% then we have a problem. So, we assume that so far, it has not happened that 51% are colluding and therefore, we are we are kind of safe. So

as I said that solving hash puzzles is probabilistic. So when you will solve the next one is actually follows the exponential distribution with that average of 10 minutes.

(Refer Slide Time: 19:56)

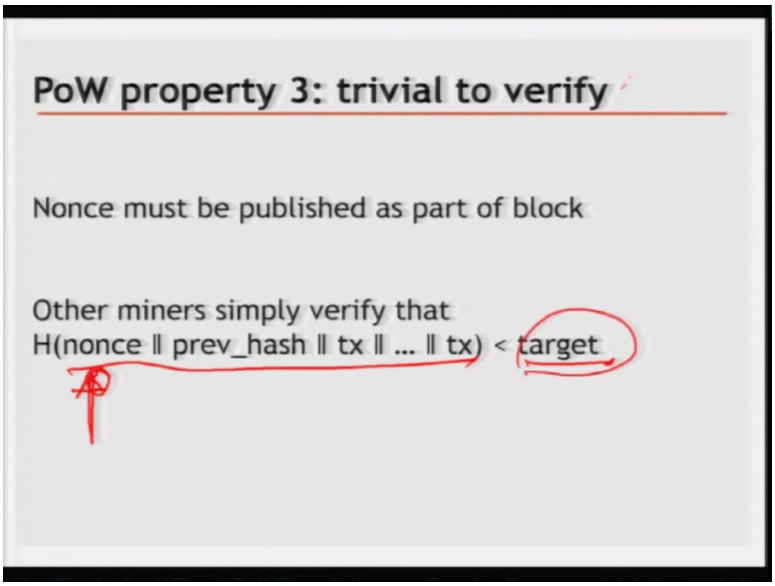


But sometimes you may need a lot more time and sometimes you can do we went faster than 10 minutes. So if it so happens that it becomes faster than 10 minutes for quite a few blocks, then we say that, the hash puzzle has become easier if people have thrown in a lot more resources. So let us go and adjust it so that it becomes again, average becomes 10 minutes. That is the idea. And for each individual miner, the meantime to find a block is 10 minutes divided by the fraction of hash power.

So if you have let us say, half of the hash powers, then your time to get this is your average will be 20 minutes, if you have one third of the hash power, then you have the average will be 30 minutes,? So therefore, the probability will accordingly also be adjusted. So if your, mean of your exponential distribution is 20, then your probability of solving one in within a fixed interval is also less than if you are more than if you are going to do it in 30.

Because the exponential distribution, the average is actually reciprocal of the lambda of the distribution is reciprocal to the mean of the corresponding exponential distribution. So your probability will be better if you are doing it in 20 minutes, if your average is 20 minutes versus if your average is 30 minutes.

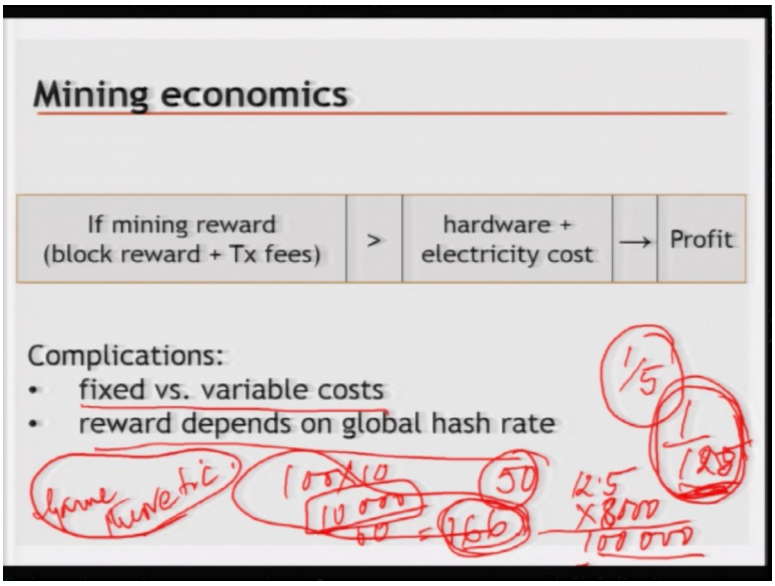
(Refer Slide Time: 21:20)



The other thing that must be true is that the, it must be trivial to verify when somebody claims that I have solved the hash puzzle for all other miners they have to verify it because before they add that block, to their copy of the blockchain, they need me to know that if it is indeed one of the winning blocks. And it should be easy because they know the target. And all they have to do is the compute one hash that is the one this nonce becomes part of the block.

So the winner has to not only broadcast his block with all the transactions and the hash of the previous block, but also has to put in the nonce in the block. So once you get that very, quickly can compute a single hash see the guy who has did the mining, he has computed probably millions of hash to in order to reach to that particular nonce. But if you are just very fine, then you have the nonce, you just plug it in. And you see whether this is below the target. And that that is why it is it is trivial to verify.

(Refer Slide Time: 22:28)



Finally, the mining economics, if I did not get enough reward, that will be my investment, that is hardware and electricity costs that are incurred throwing in so much computational resources, hardware, software, cooling costs, electricity costs, and so on. Then I would not do it so I have to have enough high probability of winning, if I am throwing in a lot of resources, so there has to be a profitability in mining now. There are a couple of things in this.

That is the fixed versus variable costs. So hardware cost is fixed, but the electricity cost is variable. So you have to accordingly do some amortization level computation, you have to see out of how many subsequent blocks you have, you usually make a block. So if your probability of being one of the winners is one over 5, then every 5 blocks, you will be making 1 block. If your probability is 1 over 100, then every 100 blocks, you will be probably making 1 block. Now, 5 blocks takes about 50 minutes, so in 50 minutes if you can win 12.5 bitcoins, and then each of them let us say cost 8000. So, you are talking about $100,000 in 50 minutes.

(Refer Slide Time: 23:48)

So, if you can win $100,000 in 50 minutes, then you have to see that to obtain one fifth of the hash power, how much hardware you have to invest in how much we will be your electricity cost over a period of 50 minutes. And our cooling costs over a period of 50 minutes, you will subtract this if your probability is 1/100, then you can only win 100,000 in 100/10 that is 10,000 minutes. So 10,000 minutes is a long time.

So that is about we divided by 60. So that is a 166 hours. So if you if you take 166 hours, so you have to see whether the hardware cost and the electricity cost that you incur in 166 hours, obviously, you have a much less hardware, so much less electricity cost and much less cooling cost. So you have to see whether that makes sense the other otherwise you will not do this.

So the; reward, whether you get the reward or not, it depends on the global hash rate. It is not only your rate, so your probability let us say is 1/100, suddenly somebody invests a lot more? Suddenly your fraction of hash power reduces. And then let us say it becomes 1/125. Then again, you have to recalculate all this? So it is reward is not only dependent on you, or what is the probability of you are getting a reward is not only dependent on you, but also what investment others are making.

So that becomes a game theoretic problem. And we will not discuss this in this class but there has been a lot of work on applying game theory to see whether the strategy of each miner forms

a Nash equilibrium or not? If they do then it makes sense that they will stick to their strategy. Otherwise, they may not stick to the strategy and shift their strategy. So we will actually end this session here.

And then in the when we come back next session, we will just basically summarize what we are discussing with respect to the Bitcoin consensus and how the consensus works and also talk about what we are going to see in the in the future in this class.

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

Lecture No. 08
Blockchain Technology and Applications

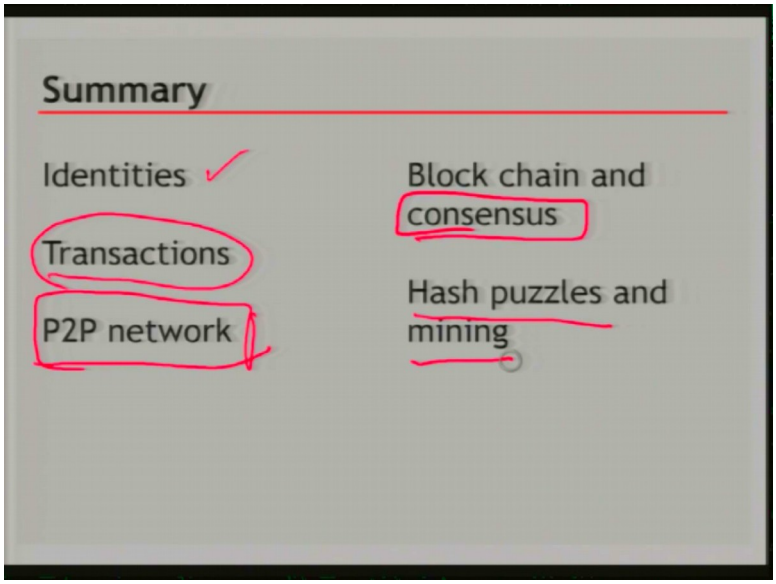
Welcome back. So, we have been discussing in the last few sessions about the bitcoin, blockchain, consensus mechanism. And we have seen that consensus in bitcoin is actually a heuristic method. It uses incentives for miners to behave properly according to rules. It also uses the, you know, flexible deadline for achieving consensus which basically leads to an eventual consistency of the blockchain on all its replicas.

It also you know allows you to dwarf any sybil attack by making the probability of making your being the being the next reward winner proportional to your global hash power. And we also saw the economics that should be the mind of the miner in terms of whether to participate in the mining process or not. When we go into the, you know, public blockchains, the private blockchains or permissioned blockchains.

We will see that most of them did not have a currency associated with them. And if they did not have a currency associated with them, a currency is not mined within the blockchain or if the currency that is mined does not have any market value. Then we have we need another kind of consensus we cannot depend on this kind of participatory consensus mechanism. So but we will we will get to that later.

When we go into the permissioned, non permission blockchain on non public blockchain and but for public blockchain, most of the public blockchains use some form of mining, which are, you know, some variant of this and so, let us understand it finally, you know putting it all together.

(Refer Slide Time: 02:23)



So, there are several concepts that we have discussed. We discussed the notion of identities in case of permission free, permission-less blockchain, what we see that identities are not tied to real world identities, your public key hash of the public key is kind of your identity. And this means, that I can create as many identities as I want. And many times the way the blockchain transaction rules are, you have to create multiple identities.

For example, if you are paying somebody some amount and you have some left, then you have to create an alternative address that is controlled by you to which you have to send the rest of the money or that is called a change address. So, every time you make a new transaction, you have to create a new identity. So, everybody will have many identities. So, in a network with a with such kind of identity, which is not tied to real world identity, there is a possibility of a sybil attack which means that somebody might create too many identities to wield more power.

Let us say in a, if there is a voting being done, to do some computation or to do leader election and so on. So therefore, what we have seen is that to dwarf the Sybil attack, the mechanism of consensus in blockchain has built in criteria that the probability of you being the winner in cases of competition to be the maker of the next block, your probability will not depend on how many identities you have, but rather it will depend on how much resources you have.

So, by creating more identities, you cannot increase your resources. So, therefore, the process is sybil attack proof, you also saw the transactions, how the transactions are represented and

you know the transactions are represented by giving transaction number inputs and outputs. And then every time you make a new transaction, you have to spend from the output of a previous transaction if that output was assigned to you.

And by using your signature, you prove that you actually are the rightful or authentic owner of that outputs destination and therefore, you are to use that amount and finally the amount should be such that the input amounts should be equal to the output amount and in case you are giving a transaction fee, the input amount should be little more than the output amount. So, therefore, not having balance will not be a problem like as we were discussing earlier that you have to check your balance.

In your account is more than your amount transaction amount that problem cannot happen here because your input some of the input should be equal to the sum of the outputs or inputs should be slightly more than the outputs in case of transaction fees. We also saw that all transactions are actually broadcast to everybody and the broadcast mechanism is through a peer to peer communication.

And what it means is that, so, bitcoin is an application level protocol, bitcoin protocol and underneath it is there is a network and above the network like TCP/IP layer, you have also another layer which is the P2P layer. So, whenever you are trying to put a transaction into the network, you basically broadcast it to all your neighbors and then your neighbors will broadcast it to their neighbors, and so on.

And then eventually some of the nodes will actually receive same transaction from multiple different sources because you are broadcasting it this way. And they will have to ignore the ones that they already have seen. And we will talk about that in the next series of lectures in the class, but that is the idea. So the P2P network is one of the, you know underlying layer of protocol on top of TCP/IP and below the bitcoin protocol.

The consensus mechanism is what we focused mostly on in this lecture, and it is very important for us because without the consensus, we cannot decide what should go into the blockchain next. And so the block that is added at any point in time should be, you know, decided based on a consensus. But we also saw that this consensus mechanism here is not

exactly like the consensus mechanism that was considered in distributed computing like in the context of distributed databases.

Because there are so many impossibility results which might make you consider abandoning the idea of building a consensus protocol in a de-centralized distributed environment like this but we saw that we actually change the model. We basically change the model in the sense that malicious nodes we assume maybe meant to behave according to rules by giving incentives.

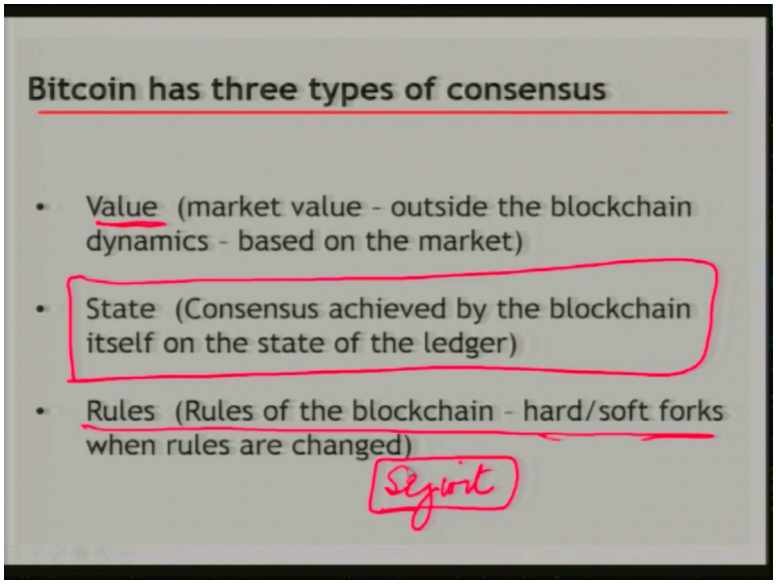
So, therefore, the whole assumption about maliciousness that is used in the old literature is a little bit changed. Second thing that we are doing is that we are not saying that consensus should happen within each round. So when a set of transactions; when each node accumulates enough transactions; they may propose a block, they solve the hash puzzle, and then they broadcast if they win.

And then multiple people may actually win at the same time or close to each other in time. And then they might broadcast there is also there might be a race condition, however, eventually be race condition will be resolved. And that is, what is the eventual consistency? So, we are allowing our consensus to be based on eventual consistency which is also different from the traditional model of a distributed consensus.

Finally they are guarantees are probabilistic, which is that the nodes the way they behave, and their possibility of broadcasting or not broadcasting is all probabilistic. So therefore, it is a randomized consensus. And all this thing together gives us the, the protocol that works in practice. And finally, the other issue the hash puzzle we have been talking about from the very beginning. And we see we saw how the hash puzzle plays important roles in the mining miners are made to solve the hash puzzles to show their proof of work.

And the proof of work we assume is actually dependent on how much computational resources you are putting in and therefore, person who solved the hash puzzle faster, has more computational resources, and therefore the probability of them winning the competition to make the next block is proportional to the computational resources that they have. So that is the kind of the summary of what we have been discussing.

(Refer Slide Time: 10:14)



Now few more general things about bitcoin. So, bitcoin actually has 3 types of consensus. And all the 3; consensus are not done through the bitcoin protocol. So, the second one here in this list is actually called a state consensus. That is, the blockchain runs the protocol, so that the state of the ledger consistent throughout all the replicas. So that is the consensus that we try to achieve through this protocol.

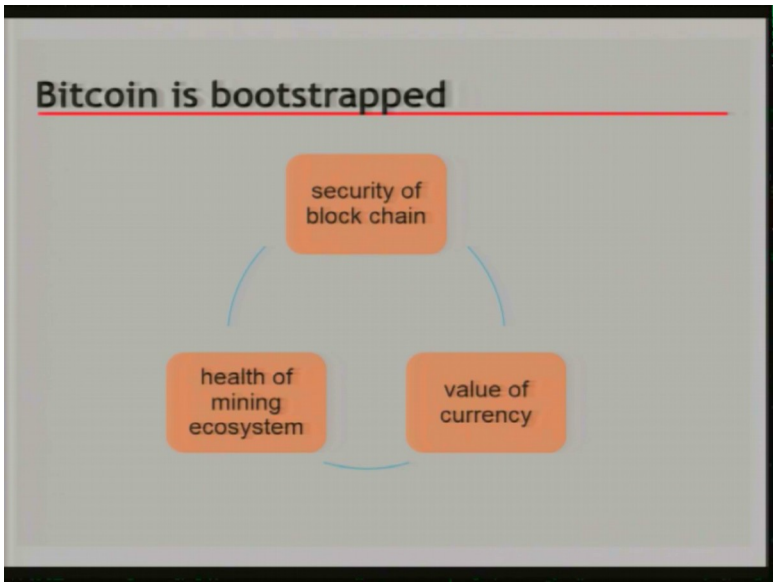
And that is what we discussed all along in the last few sessions. But there is also a value consensus that is the bitcoins market value. This market value is not now has nothing to do with the bitcoin protocol. It has to do with lots of extraneous factors such as you know, what is the news in the market or how the investors seem they bitcoin. Bitcoin going up and down up or down in value.

And therefore, this is also done through a distributed process there is no central authority or bank that decides the value of the market value of the bitcoin, but, actually it is decided on by a decentralized process based on demand and supply and then there is a there are rules. So, we said that there are certain rules in bitcoin, for example, how the transaction validity plays out how the hash puzzle works out, what is the difficulty of the hash puzzle.

And all these things are actually the rules and this rule should be played along by everybody. So, that is another consensus. And then if a large number of nodes decide to play by a different rule, then there is something called forking in the blockchain. So, you will see that 2 years ago, there was a change in the block structure and this is called the Segwit. And because of segwit, there was actually a fork, in the hard fork in the bitcoin.

And there was a block cache version of bitcoin that forked out of the blockchain. So this; so, we worried about the second type that is the ledger consensus, but the other consensus are also important in the entire scheme of things. The other issue is that we want to point out is that bitcoin is bootstrapped.

(Refer Slide Time: 12:47)



Now, what do I mean by this? So when the bitcoin came into existence, very few people knew about it, only people in crypto community knew about it and the amount of for mining resources was you know desktops and laptops and so on, and therefore be anybody and everybody had almost equal chance of winning the next block. And what that meant is that nobody had the ability to overpower the existing chain and creator.

You know, fork or create a different chain than the legitimate consensus chain. And that has changed drastically over time as some people have put in more resources for bitcoin mining, and therefore the health of the mining may be deteriorating. And as we get news of the health of the mining system, being worse, for example, if there is news that China is going to nationalize all its blockchain mining companies and put them, consolidate them.

That would mean that investors will immediately panic that there might be a 51% hash power within that consolidated company. And therefore, the blockchain can be easily subverted and your old transactions may become invalid, people will take out their investment from bitcoin. And what that would do is that the price of the bitcoin the value of the currency will fall. So

health of the mining ecosystem and the value of the currency are related to each other, the how they are related to each other.

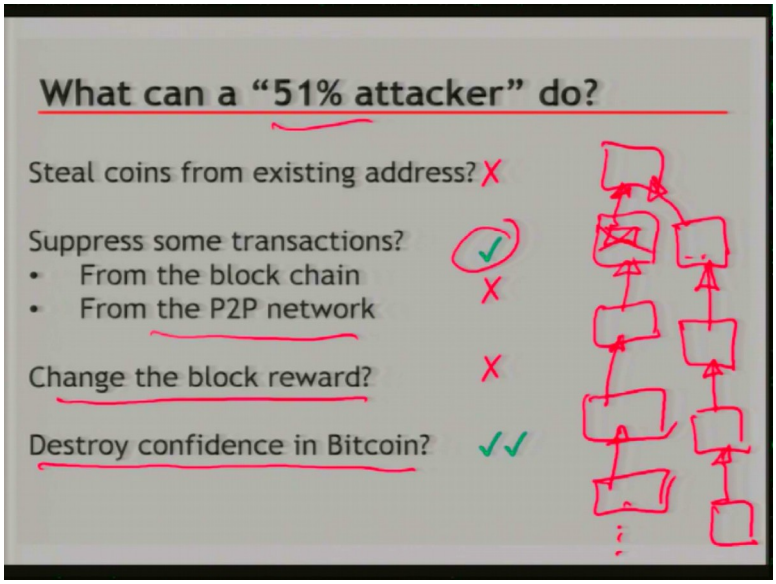
So, we just saw we just described how the health of the mining system influences the value of the currency. On the other hand, the value of the currency also influences the health of the mining system, because if the value is very small, people will not invest in computational resources to do mining, very few people will do mining and therefore there will be no competition to create the mining.

And therefore lesser mining resources, there are the chances of somebody actually among the risks of the miners becoming 51% becomes higher. So, therefore, the value of currency falls means that a lot of people will leave the mining business. The other thing is the security of the blockchain if the security of the blockchain which is dependent on the health of the mining system we have been saying this that if the mining ecosystem is not healthy, then you might see 51% attacks and maybe other kinds of attacks.

And therefore, you will see that the blockchain security is compromised once the security compromises the value falls, because investors will fly because they will see that there is no value in the in the currency there is no security so anytime the bitcoin, old transactions become invalid and so on, nobody is going to invest. So as the value of the currency falls, the more miners will flee and then the security will decrease.

So, this is kind of a cycle so in that sense that all these 3 things kind of is must be in equilibrium in order for the bitcoin to retain its value, retain its security of the ecosystem and have enough miners to mine bitcoin. So that is, the point that is discussed here.

(Refer Slide Time: 16:22)



Now, one more thing is that we keep hearing in the news about 51% attack possibility. So what can a 51% attack do? 51% attack can actually do a lot of things but not everything. For example, a 51% attacker cannot steal point from an existing address, because that existing address might have coins and it received those coins through some approach as output of a previous transaction and only way to steal those coins is to know the private key of that addressing.

If you did not know that private key, you can never redeemed the coin. So 51% attacker or not, it is cryptographically protected. It is not based on the consensus, the 51% attacker can only influence the consensus or how the blockchain develops, but it cannot subvert cryptography. Second thing is that can it suppress some transactions from the blockchain? Yes, because it can actually if it has 51% power, it will start developing another chain.

So, if this is the original chain where your transaction was there, and then it will start and then you actually got 6 confirmations. So you are now or got 5 confirmations and you are waiting for a one more confirmation before you are you can redeem. At that point, the 51% attacker can come and start building here, another branch of the chain and since he has more hash power, he keeps building more and more of these.

And others also get fooled and start building on this and this becomes longer. So then your transaction becomes no longer part of the permanent record. And therefore, you are you cannot redeem that money. So, that is a possibility. And that possibility can also happen if you had six confirmations if 51% attacker attains let us say 90% so in that case, he can really

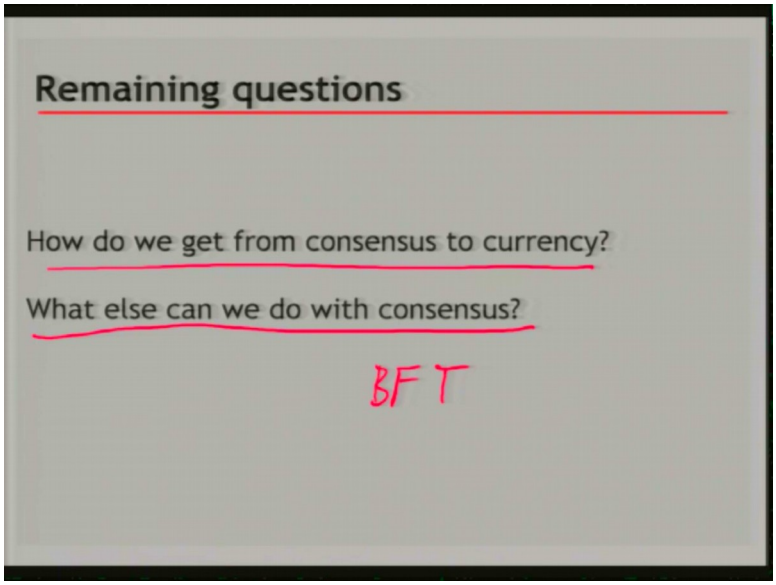
build every 10 times, 9 blocks will be built by him, and therefore, he can very quickly surpass even the 6 length long confirmation.

So that is a real possibility depending on how much above 51% he is and even with 51% there is a probability that you will make more than half the blocks and therefore he can surpass the existing chain. But he cannot surprise the transaction when the transaction is being broadcast into the P2P network that requests an attack on the P2P network by network cyber attack rather than on the consensus process or on the block making or any other cryptographic technique.

So that he cannot do carry change the block reward, no he cannot change the block reward because block reward is part of the rule. And therefore, you know even if you are 51% you cannot make blocks which gives you more reward because if you do give yourself more reward, then others will not validate that block and therefore, you know, you will not be able to do this. Now, what you can do very well is the, destroy the confidence in the bitcoin ecosystem.

And therefore, the value of the bitcoin will fall miners will flee and the whole bitcoin infrastructure will collapse. And that is what we were trying to actually say in the previous one, that all of these things should be in equilibrium so that the entire ecosystem does not collapse. The value collapses the miners flee and therefore, you know transactions cannot be cannot be put in into the block or securely and things like that.

(Refer Slide Time: 20:20)



So, the remaining questions that will take up next in the next classes is that how do you get from consensus to currency? And so, because we discussed consensus and we discussed, you know, creation of currency, but eventually there has to be a mechanism and how the way the consensus gives us currency, and that is called a Bitcoin mechanism. And we will be taking up bitcoin mechanism as our next topic.

And the other question is, what else can we do with consensus? And this is the, question that we will be pondering even after when we talk about other blockchain systems, all of which will require some form of consensus. And so we will look at what other consensus mechanisms are there we see in some cases, we directly do Byzantine fault tolerant consensus, or randomized, Byzantine fault tolerant consensus.

Sometimes we do other techniques but eventually we have to have some form of consensus in order for creating permanence and verifiability in a decentralized system. So that is basically the end of this and we will be actually talking about the Bitcoin mechanics, how the transactions look like what are the different things that go into a transaction, and then how the transactions, how the blocks look like.

And how the transactions are stored in the block what kind of data structure it is stored in now you might say that if we are not interested in crypto currency, why are we going to go into that much detail of Bitcoin mechanics? The reason is that many of these concepts that are being used. There are actually pretty general so many of the bitcoins, many of the blockchains actually use those same techniques.

So, for example, the way the transactions are stored inside the inside a block is called a Merkle tree. So we discussed Merkle tree before, so all the transactions are not stored like a, like an in an array or something they are actually put into a Merkle tree. And we saw that the reason why we use Merkle tree is because the searching for a transaction is faster. So the, that is one of the lesson from Bitcoin blockchains block structure.

That you if you have a lot of transactions, you might you might be better of putting them in a Merkle tree. And then we will see that in even later we will see that in ethereum, this Merkle tree has been further optimized. So that is the one thing. The other thing that we will look into

is that in a Bitcoin the, you did not just put your signature and you did not just put your destination address, but you actually write some scripts.

And those scripts are executed by everybody who wants to validate that transaction. And this scripting execution basically make allows you to add the script. How did that script language how much power that scripting language has determines how many clever way clever types of transactions you can do? So we will see that there are many type of transactions other than just kind of transactions you have seen so far.

For example, there are multi SIG transactions which allows you to jointly do something. And then multi SIG allows you to do further things like an escrow transaction. And then you can also do what is called micro payments. So there are many different ways of doing transactions cleverly with the help of the scripting language and that is called a Bitcoin scripting language. But you will see that even with the bitcoin scripting language, you have a very limited power, you can only do certain things.

And that led people to think that why cannot we write much cleverer transactions and therefore, they started thinking about a new language for writing clever, complex transactions and that led to the idea of smart contracts. So will not get into smart contracts until we get to ethereum but smart contracts are basically pretty generic language compared to Bitcoin scripting language to write complex rules for transacting.

And that means that we evolve from blockchain 1.0 to blockchain 2.0 as we introduce smart contracts. So, in order to understand smart contracts, we have to first look at bitcoin scripting language and understand its inadequacy which led to the smart contracts. That is another reason why we want to go into the Bitcoin mechanics in a more details, even though I am like totally opposed to the crypto currency.

Because the unless we see what exists and what are its pitfalls, we cannot evolve to the next good thing and that is another thing that we will study in the bitcoin mechanics lectures after we are done with the bitcoin mechanics lecture, we will move to the ethereum. And we will look at smart contracts. But in this course will not be actually teaching you how to write new smart contracts. We will see some examples you will get some ideas, but the point we try to.

You know, put across to you is not to make you become a Bitcoin or ethereum, expert or programmer of smart contracts but rather conceptually understand what the blockchains are, what the dynamics are, and how you can use blockchain for many applications. And after we are done with the smart contracts and in ethereum context, we will move to a permission blockchain which is hyperledger.

And we will see how hyper ledger uses smart contracts. How hyper ledger does consensus. So, consensus in ethereum is very similar to that of blockchain, Bitcoin blockchain, but consensus is hyper ledger is entirely different. So, because of that, we have to understand how the consensus is done in hyperledger and what are the other possibilities in hyperledger to do the consensus.

And interestingly, in hyperledger, they have a pluggable consensus so you can write your own consensus and plug it into the hyperledger. Hyperledger is an open source system, open low open source blockchain so ethereum and bitcoin as well. So, then we will see, what are the other applications that people are using and why are they not using one of these previous blockchains and developing their own.

So for that we will look at least one for specifically designed for IOT systems and it is called Iota and then we will look at one for the financial, you know, players, and that is called Corda. And then if we have time, we will see a few other blockchains. And then we will look at least 1 or 2 applications like land record on a blockchain or medical information on a blockchain this kind of an application. So, at the end, you should have a very good idea about the underlying theory of the blockchains.

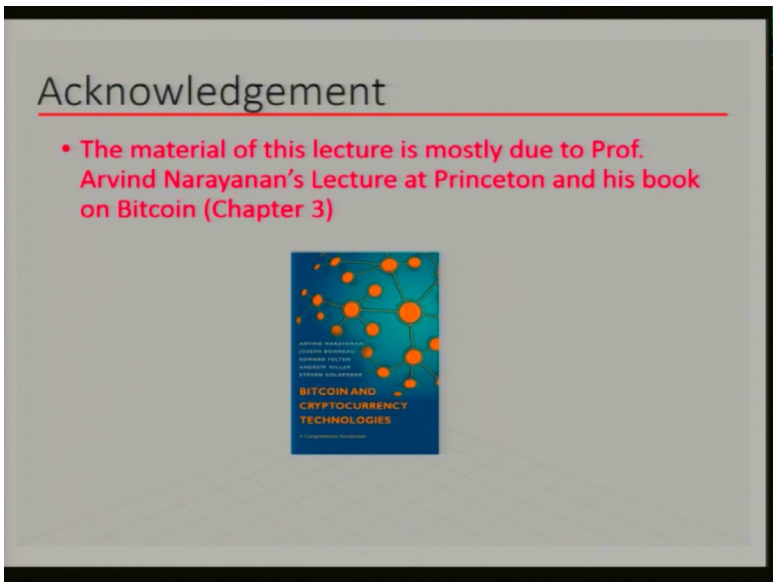
And then, what are the different types of blockchain the evolution of blockchain technology and where it is applicable and how to apply them and then eventually choose choosing which blockchain is your, most applicable for, your application, and whether blockchain is at all applicable for your application. So this is the game plan here in the rest of the course. So, we will see you next time with the Bitcoin mechanics.

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

Lecture No. 09
Blockchain Technology and Applications

Welcome back to another session of blockchain technology and applications. In the last few sessions we have learned a lot about bitcoin consensus mechanism. But we have not seen the details of transactions and how the transactions are validated by the different participants in the blockchain. And we also have not seen the how the blocks are validated. So, today, we will talk about the blockchain, bitcoin blockchain mechanics.

(Refer Slide Time: 00:53)

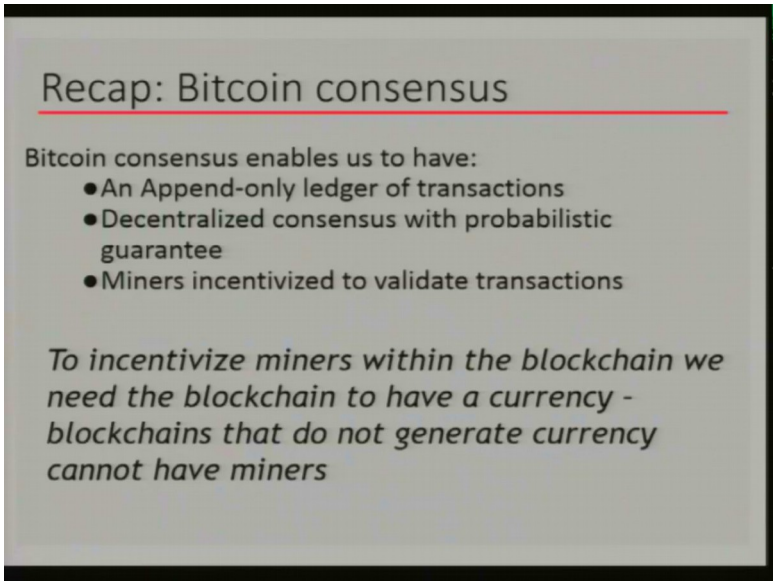


So, today's talk is entirely taken from the lectures and the book by Professor Arvind Narayanan of Princeton University. And it is a very interesting book and if you are interested in learning all about Bitcoin blockchain and other alternative crypto currencies, then this is a very good book, I would recommend that to you.

(Refer Slide Time: 01:12)

But we will be taking only selected parts of the book that is necessary for you to understand the basic mechanics of the Bitcoin blockchain. So the consensus that we have seen allows us to have an append only ledger of transactions. And the consensus is decentralized.

(Refer Slide Time: 01:44)



It is a distributed algorithm, but we overcome the impossibility results in the presence of adversaries or in the presence of malicious users. Because we are using probabilistic guarantee and eventual consistency also we are using incentive mechanism and incentive mechanism has 2 purposes. One is why would I do go and do the validation of your transaction? So that is solved by giving incentives.

The other problem, as we discussed is that there might be malicious adversaries as participants because this is a sort of pseudo anonymous network and it is very easy to enter

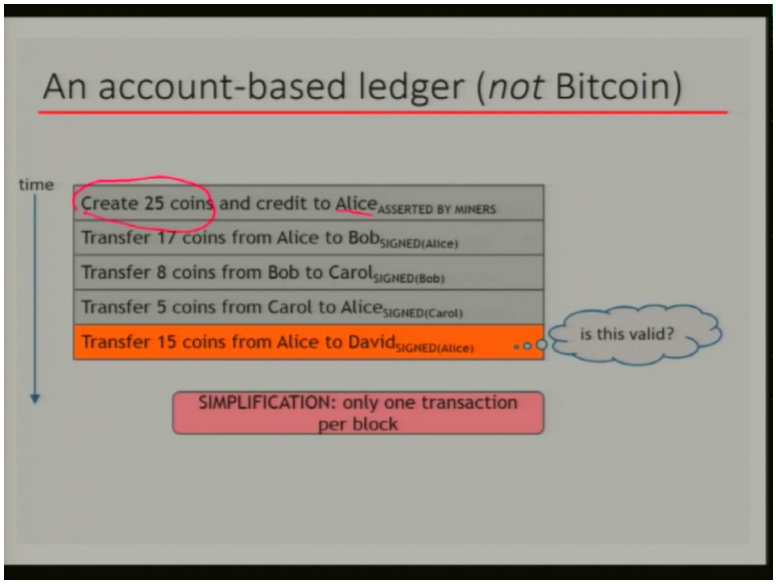
the network. So therefore, we use some mechanism for incentivizing the users to behave according to rules. So, one thing that you must note here is that Bitcoin blockchain is basically a blockchain for crypto currency creation and crypto currency transactions.

So if you have a blockchain and we will see many blockchains that did not have a native crypto currency, and if they did not have native crypto currency, such incentive mechanism would not be available to us and there, we have to worry about other ways to solve the problem of validating transactions, validating blocks and also creating the blocks that does not make the blockchain go in all possible directions and keep the consensus branch most longest branch. And therefore, make the blockchain integrity intact.

(Refer Slide Time: 03:30)

So, what we mean by bitcoin mechanics. Bitcoin mechanics mostly is about how transactions are done, what are the data structures that are used for denoting a transaction and what are their purposes? Why do we need those kind of entries in the blockchain transaction structure and also how people, the miners that are or other members of the participant other participants validate the transactions and also we will look at the blocks and block validation. So first let us look at the bitcoin transactions.

(Refer Slide Time: 04:13)



So first thing one would think having been used to the accounts based ledger is that one might consider an account based transaction and an account based transaction will look like this that suppose I have a transaction that creates 25 coins and I say that who it goes to, and this is asserted by the miners who create this 25 coins. But then let us say Alice transfers 17 coins to Bob and science digitally sign the transaction.

And then Bob transactions transacts 8 coins to Carol and signed by Bob, and then 5 coins are given by Carol, back to Alice. Now how many coins now Alice possess. So there are 25 coins in the beginning which Alice got, she gives 17 to Bob, so she had 8, and then the next transaction does not involve her. But then the transaction after that gives 5 coins to her so she had 8 and now she has 5 more so she has 13.

But in order to know that Alice now has 13 we have to go back to the for very first transaction and work our way down in order to see where Alice got a positive inflow and where she got a you know outflow. And then we can also consider these as if these are the blocks. So now, let us say at this point, Alice wants to transfer 15 coins to David, so all the participants need to change the validity of this.

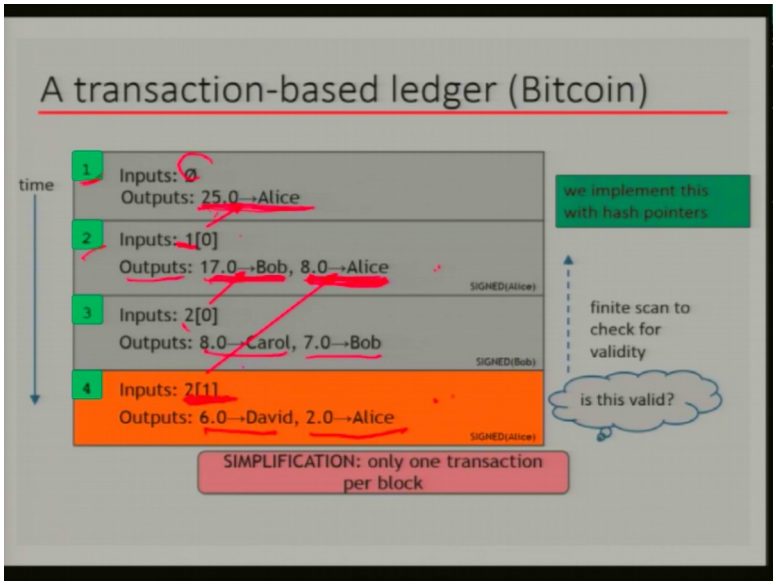
And in order to check the validity, they may have to go back to the first transaction where Alice at 25, then they, find that next transaction, Alice gives 17. So she has 8, then next one she does, it does not involve her, and the next one, she gets 5, so she has 13. So she cannot give 15 coins to Alice. So it is an invalid transaction. So in order to decide that, we have to go through all the transactions that happened before.

And if you consider each of this transaction as a block, we are basically going all the way to the genesis block to find out whether a transaction is valid. Now, this is going to be very difficult. Now, as we saw in our prototype blockchain that we actually discussed and given size on that, in that we were cashing this balance information. So every time we validate a transaction, we update the balance and that is also possible to do.

But that requires a lot of more work for every participant because every participant needs to validate these transactions. So therefore, if they have to cache all this information, then they have to do a lot of work. And then there are a few other complications because sometimes the blockchain will diverged a little bit and then they converge back to the main consensus chain. So, therefore, those things becomes a little bit non deterministic.

So that is not a very good solution to do. In our example, that was left to us exercise. There, we were only one party that was running the entire program, it was not replicated all over the places. So there was no non determinism involved. But when you have, you know, 1000s of participants involved and each of them are independently doing all this validation of transactions and also keeping track of the balance there would be some inconsistency creeping in and therefore, there could be some problem. So, therefore, this is not what bitcoin does.

(Refer Slide Time: 08:11)



So, what bitcoin does is actually a transaction based ledger. So they only keep track of transactions not the account balances. So, the first transaction for example, this is a genesis

transaction and what is happening is that there is no input to this transaction, but the output says that give 25 coins to Alice. Now, at this point, the next transaction only Alice can use. Now, what Alice has to do now is that she has to refer to the transaction that gave her the 25 coins.

So, transactions are numbered so, these are transaction IDs. So, she refers to the transaction ID 1 and then in this case there are only there is only 1 output so the outputs are and inputs are basically can be thought of as Alice. So, she refers to the first entry of the output of transaction 1. So this means I am inputting for from transaction 1 and only way I can input something from the transaction 1 is from the outputs of the transaction.

So the output number is 0, so there was only one output here. So, output numbers 0 is 25 coins going to Alice. So, and then she says the, my output has 2 entries. So, this is the entry number 0 and this is the entry number one where in the entry number 0. I give 17 to Bob and give 8 to myself to a different address. Remember the change address concept. So, Alice creates a change address which is different from her original address.

And then she gives that money to herself because otherwise there was a replay attack that we discussed before. So, now the next time there is a transaction. So, in this case Bob is making the transaction. So, Bob is refers to transaction number 2 and entry numbers 0 in the outputs of the transaction number 2, which basically gave him gave him 17 coins. So out of the 17 coins he gives 8 to Carol and then gives to one of his change addresses 7 coins.

So therefore, now Bob has given a casual 8 and himself back 7. Now in this one Alice again wants to make a transaction. Now she has transaction she got money in transaction 2 she cannot use transaction one anymore, because transaction 1 output has been spent. So transaction 2 is the only a transaction where she got money. So from herself only but she got money there. So she refers to the entry number 1 of the outputs of transaction 2.

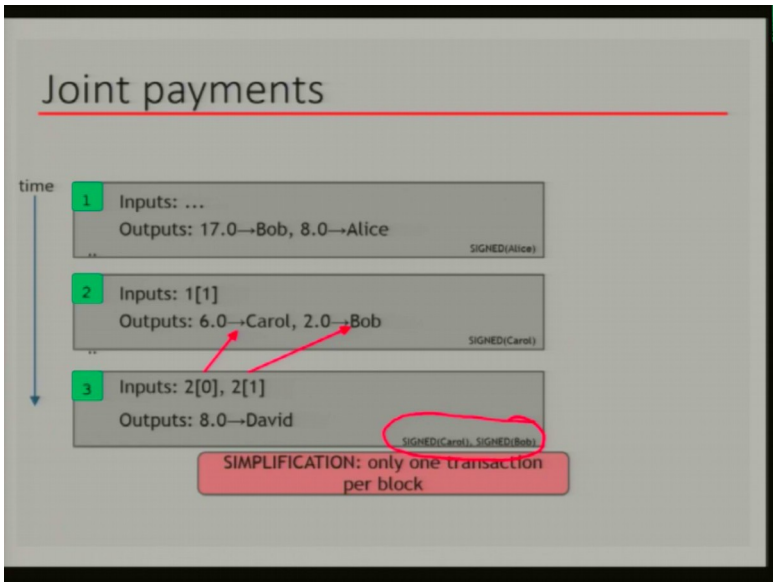
So that that is indexed with one, and there she gives 6 to David and then to change address. So this is the way bitcoin does this transaction keeps track of the transactions. And note that in this case, we want to see whether this transaction is valid, all we need to do is to look at the transactions she refers to as a source transaction those transaction is this. And we see that in that source transaction in the output number 1, she got 8.

So that is all we need to check. So now you see that we did not have to go all the way back to all the transactions, we also have to check whether Alice has used this transaction 2 index 1 anywhere between this transaction and this transaction that also has to be checked. So therefore we have to only check a fraction of the transactions and not all the transactions all the way up to the Genesis transaction.

So that is actually the Genesis, it is not called the Genesis transaction. It is called the coinbase transaction. So coinbase transaction are transactions in which we create some coins, and those transactions did not require any inputs, coins are created in those transactions. So those are called coin based transactions. I was calling it intentionally Genesis transactions but it is not really a Genesis transaction. Genesis block is a concept not Genesis transaction.

And Genesis block means the first block of the blockchain in which the blockchain started. So we will talk about that later. But what we also do is that we create hash pointers to these different transactions so that we can easily trace back when somebody refers to a transaction, we can quickly refer back to the transaction that we are interested in.

(Refer Slide Time: 12:45)



Now, other thing we can do with this mechanism is that suppose I have multiple change addresses on many transactions. I got different number of coins or same number of coins, it does not matter. I got some points on multiple chain of change addresses. But it is very difficult for me to keep track of all those change addresses and how much I have in all those.

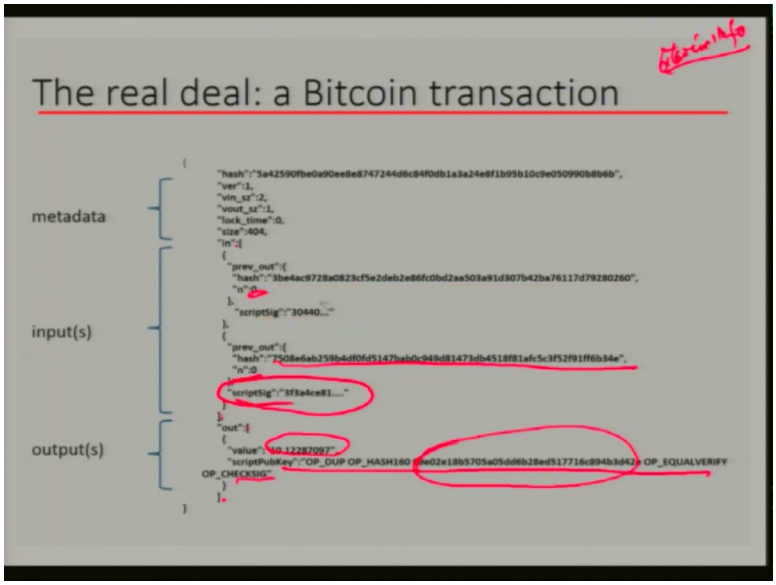
So time to time I may want to merge those change addresses, the contents of those change addresses into a single address so that that is what is called a margin.

And that is what we can do using like two input transaction so one from here and one from here. So, you see that the Bob can consolidate so here Bob got 17 in this transaction and Bob got 2 in this transaction. So she he should ideally have 19. So he actually can create that by putting in a transaction where he refers to transaction number 1, output 0 and transaction number 2, output 1 and then she says that give 19 to me.

Remember requirements in the input transactions, these are called the input transactions to this transaction and these are this is an output of this transaction. So this is an input of this transaction, is an input of this transaction and this is the output. So output can be can be also multiple and input can be also multiple. And that is what we are seeing here. So now we can also do joint payments. That is if I did not have enough coins myself, and but I am under friend together decide that we will be using our coins together to buy something, we are allowed to do that.

So the joint transaction looks like this. So let us say I have 2 transactions here. Carol has 6 Bob has to and they together decide to give to David so they can put in a joint transaction. But remember that they have to both sign this transaction because Bob or Carol, themselves by themselves did not have the right to spend both of these outputs. So they have to do a joint signing we will see how that works in a little bit.

(Refer Slide Time: 15:03)



Now in reality, so, we give you the schematics of what the transactions look like when in reality this is what happens this is the structure of a transaction and if you actually go into like you know some real browser this bitcoin.info for example, and then you can browse transactions and you will see things like this. So, what the transaction in the transaction structure what we have are the following.

So, we have metadata. So, this is the metadata this is the data about the transaction, this is these are the inputs. So you see that the input is an array, so this array is denoted by this and finished by this and then output is also an array in this case, I this in this particular transaction, there is only 1 output and I see there are 2 inputs. So in the Meta data, what we have is the hash of the transaction.

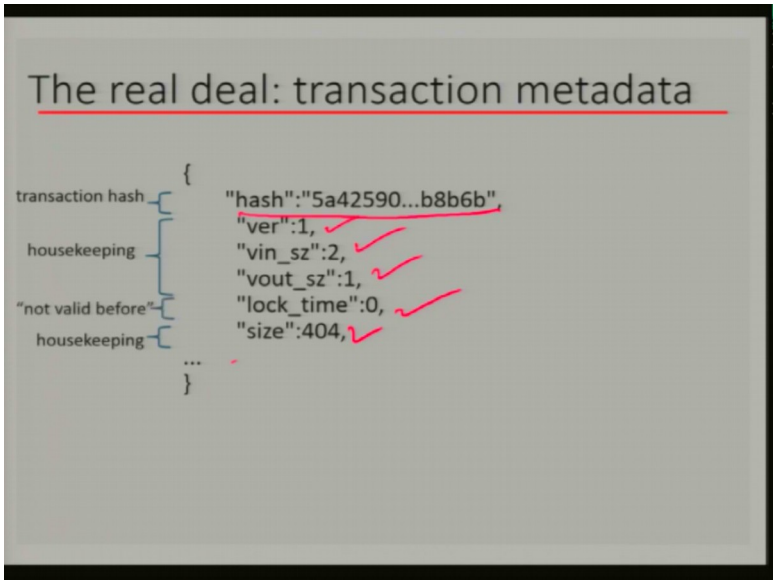
And then we have the version. This is the Bitcoin version. This says how many inputs are there. So this is the input size, this is the output size. So this is 2 and 1, there is something called lock time and we will discuss very soon. What lock time means so for now, we can ignore it accept the fact that lock time zero means this transaction can be done immediately. So that is what lock time zero means, the size of the transaction.

And then in the input, we have the hash of the transaction that is the input. And n = 0 here means that this is the, zeroth output of the transaction. And then there is a signature. Now, how this signature is done. We will we will defer that discussion for later a little bit later, but this is kind of like a signature you can first now assume, similarly the second output also has the transaction hash for the previous second previous transaction.

And then this is the, zeroth output of the transaction and here is sort of the signature in the output, you say, what is the value of the value that you are sending. And this basically says the address of the person whose account is getting the funding the getting the money. Now, here, you see that there is a lot of things, not only just account number, the account number is basically somewhere here.

But the rest of it seems like, you know, sort of like an assembly program. And we will see what that is in a little bit. So that is what the transaction structure looks like. So you get some idea about what this looks like.

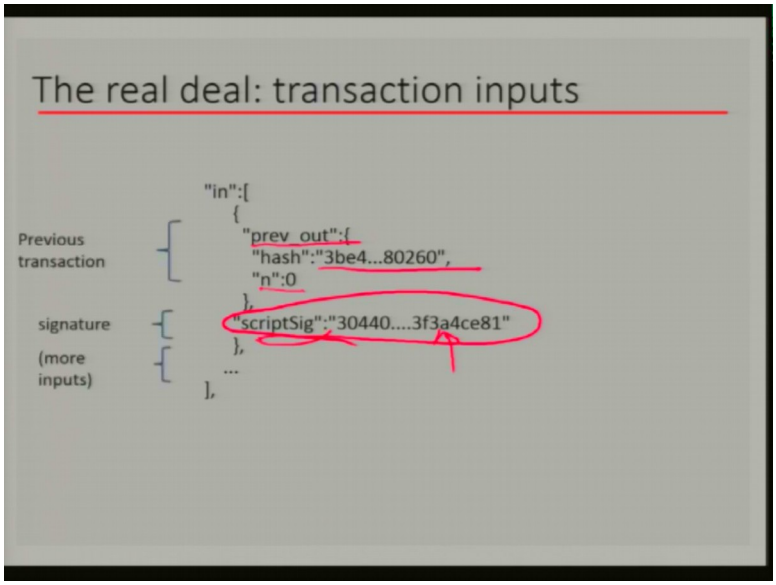
(Refer Slide Time: 17:46)



Now, let us look at this metadata and other inputs and outputs in a little bit detail. So, as I said that the metadata part has hash of the transaction, the version number, the size of the input array, the size of the output array, the lock time zero means it is an immediate transaction and the size of the transaction and there may be some other data that might be there. So that is a Meta data. So all of you know Meta data means that it is a data about the data.

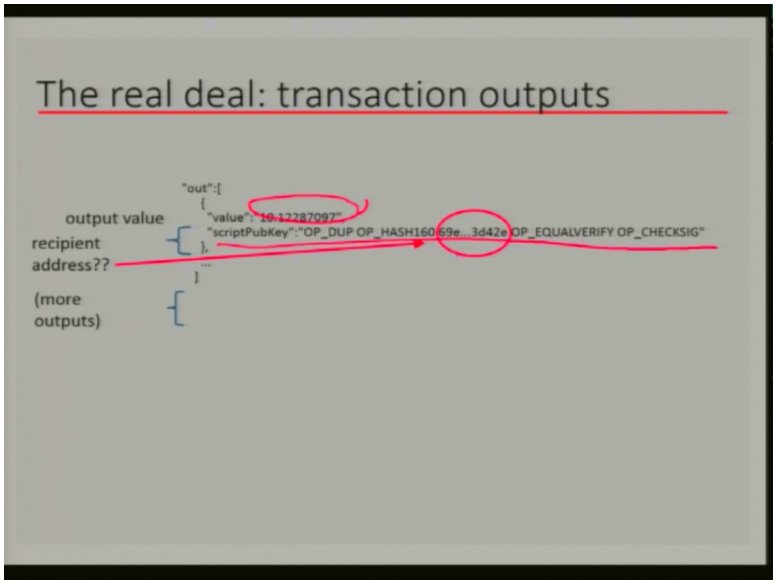
So this is the data about the transaction. So this is what kind of says a housekeeping information. And this is the transaction hash. And lock time gives you when the transaction becomes valid, zero means it becomes valid immediately.

(Refer Slide Time: 18:38)



The input looks like this. So you have the previous output transactions hash here. So and then this is the index of the output of the transaction. And here is the signature. And we will we will see what that means. This is actually a public key. And this is a, you know, script that we will understand in a little bit.

(Refer Slide Time: 19:05)



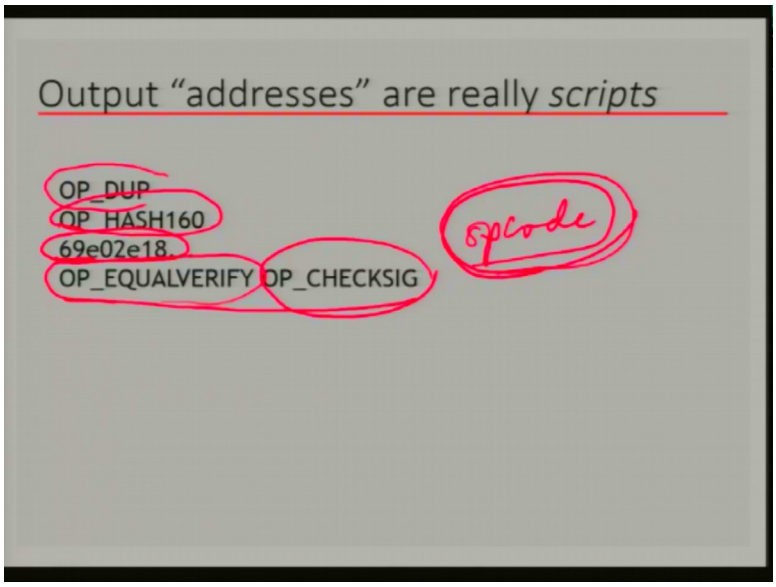
So the output also has the value of the output, how much how many coins are going and, here it says who is going to be created for that amount. And now, as we said that this kind of looks strange, but will very soon understand what it means. And this part of that is actually a hash value is the account number, but we will understand very soon.

(Refer Slide Time: 19:34)

So bitcoin scripts is what those things are so those were mysterious things that you saw, like what scripting and in our in the output it says script pubkey, and in the input, it says,

scriptsig. These are actually what we call bitcoin scripts. So bitcoin script is actually kind of like an assembly program, except that assembly programs are meant for real Hardware machines, but in this case this script language actually has an idealized machine on which it is supposed to be supposed to be executed. And this is a stack based machine on which this is executed. So let us see how that works.

(Refer Slide Time: 20:16)



So, first of all, like any assembly code, it has top codes. So it has op_dup, for example, is not is an opcode which says the duplicate something that is on top of a stack, so we have a stack based model of computation. So, data remain data is pushed into the stack. And then at that point, if we get this instruction op_dup, then we create a duplication of the top of the stack on top. So if there is x on the top of the stack at the time.

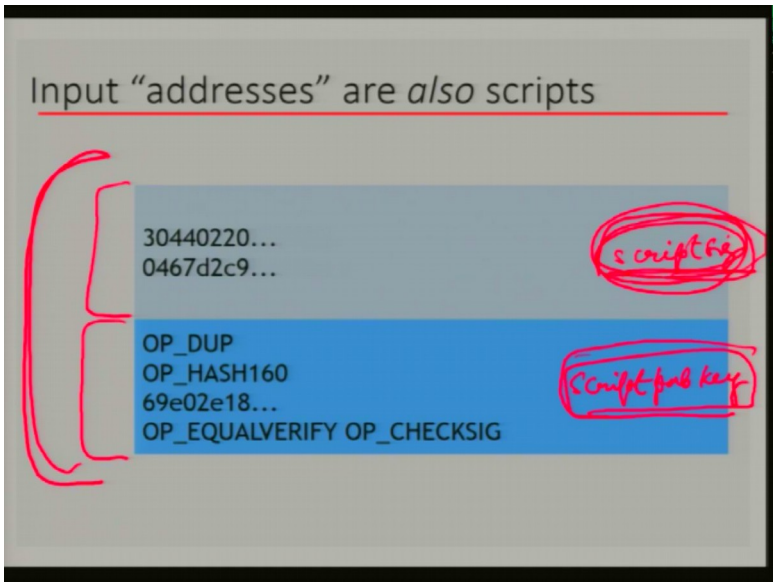
I encountered this instruction, then I will write another X on top of it op_hash basically hashes, the whatever on the top of the stack, so, it basically does a 2 layer hashing first it does SHA256 and then it does a 160 bit hash the output is a 160 bit hash. And here is some data. We saw in this case, we said that this was actually a hash value, but it could be any data. And then this says, op_equalverify op_checksig.

So, op_equalverify verifies that the top 2 entries on the stack are equal. And then op_checksig basically checks whether the signature checks out. So if you have the signature, and if you have the public key, then if you apply op_checksig, then it will verify that the signature is according to the is decipherable with the public key that means it is it was created by the private key have the corresponding public key.

So these are called opcodes of the language. So, what we are seeing here is that there is a language and this language is called a Bitcoin script language and this language is very simple and it has the semantics of this language is stack based which means that we will assume that way the programs written in this language are executed is by reading one instruction and then if the instruction is data, then push it onto the stack.

If it is an actual instruction opcode then we take the things on top of the stack, maybe the only the top entry or top 2 entry or top 3 entry depending on the instruction and then we execute on that data that instruction and then we whatever it is the result we remove the entries that on which you apply the apply the instruction and then you push the result on top of the stack if there is a result. So that is how the semantics of this language works.

(Refer Slide Time: 23:05)



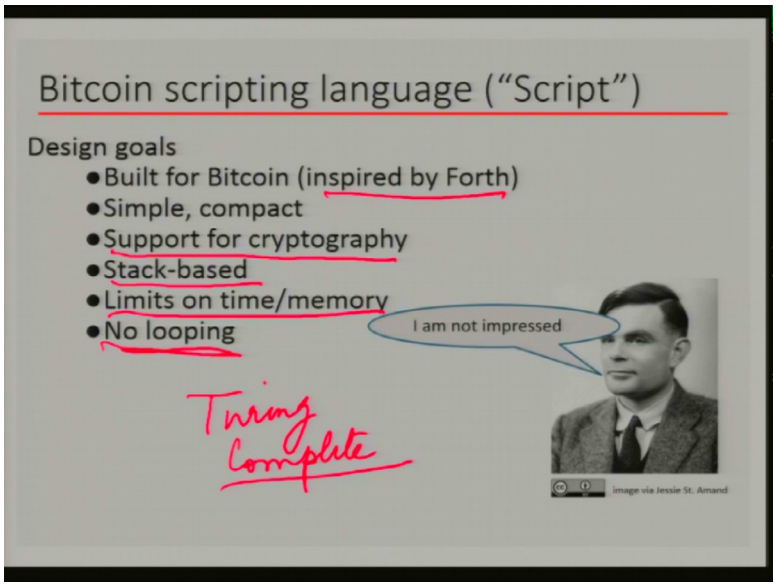
So it turns out that when we say scriptsig that actually is a script. So we would think that in the input transaction, so when I use a transaction as my input, so what I am doing is that I am looking at a previous transaction in which I was given some money. And I am going to sign that the thing that I am now signing to use that money, so it is called redeeming the input transaction, the output of the previous transaction.

So to redeem, I need to give a signature but instead of a plain signature, I give a script. Similarly, in the output, I saw that I am supposed to only say, whose account which should go to, but instead of giving just the account number as a hash of a public key, I have to write a

script and that is called a Script pub key. So we will see how this, this thing works. And, then whoever is going to validate the transaction has to put script on top of this script.

So script is basically a set of instructions and data values. And this is also a set of instruction and data value. So I concatenate them. So I have first these instructions, and then I have this instructions, I get a longer script. And then I have to execute that script. And in a stack based system, and when I execute, if the execution goes to well, then it is all fine. The transaction validates if the execution throws an error that means something did not work out. That means either this script has a wrong information or this script has the wrong information, and therefore the transaction is invalidated. So that is the idea of this scripts.

(Refer Slide Time: 24:52)



So the goal of the bitcoin scripting language is actually to have a simple, compact way of validating the transactions and we want to keep the machinery to validate the transactions. Very, very simple. That is why the language is also very, very simple. It has instructions to support cryptography. So it has single instruction to compute a hash, it has single instruction to validate a signature. So this is actually in that sense, the instructions are pretty powerful, but it is stack based.

And therefore an every instruction is executed in just once. And therefore the time and memory is basically linear in the size of this script. And therefore it is pretty limited. Because we did not want to see the scripts are written by somebody else, because when I are validating a transaction, the transaction scriptsig, or scriptpubkey, is written by the person who actually put the transaction in now is the malicious user.

And he had a time consuming instruction there, then you all the validators will actually be getting into a huge time consuming computation and they will delay or validating all other transactions. So, we want the language such that it is not possible to get into an infinite loop or very, expensive computation. So, this language has no loops in the in its structure and therefore, it is not turing complete.

So, Turing complete languages like all programming languages that we normally use are called turing complete languages. So, if you have taken a course in theory of computation, you probably know what that means, but turing complete languages can design or can express very, complex functions, whereas, this kind of a very, simple language With no looping cannot express any arbitrary functions and therefore, and that is by design.

So that we can transaction validation cannot be attacked by a malicious user by putting in a transaction where he puts a script which goes into an infinite loop and things like that. And it is inspired by previously available language called forth. We did not need to worry about that language, but we have to understand the very basics of this language. Now, why are we talking about this language that come when we come back? We will talk about that.

And we will be actually discussing the concept of smart contracts in this context and then you will understand why even though we are not interested in cryptocurrency, we are talking about bitcoin scripts. So, we will come back.

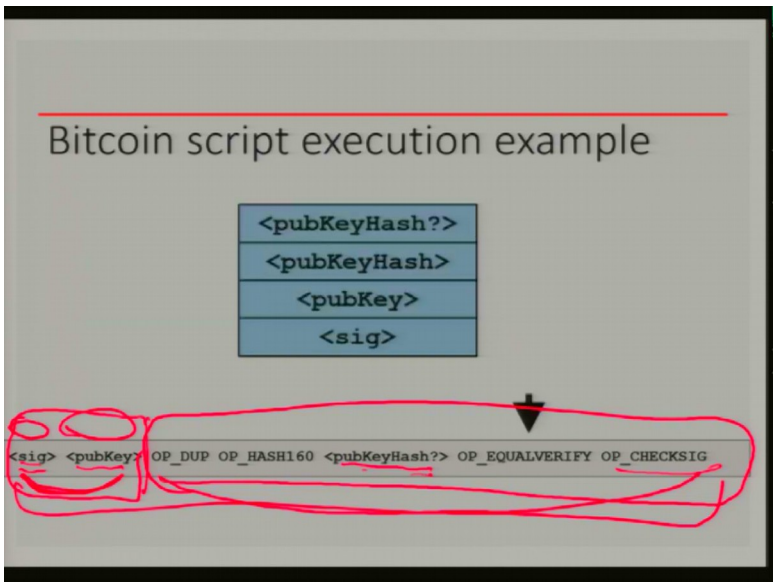
Introduction to Blockchain Technology and Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture No. 10
Blockchain Technology and Applications

Welcome back. So we were discussing bitcoin scripting language and we ended by saying that bitcoin scripting language is not turing complete. And what that means is that we cannot express arbitrary functionality with bitcoin scripting language, very, simple things can be only expressed with such a, you know, restricted language. Now, what happened is that we were when the users of bitcoin saw that, the limitations of this.

Then the Ethereum guys came in, and they said that we need a much more expressive language. So they created a new language called solidity. And in ethereum, they started using solidity to write this kind of complex functions. And those are called smart contracts. So smart contract, the bitcoin scripts are actually precursors of smart contract.

(Refer Slide Time: 01:13)



And we, will see in a few within a few slides, that the bitcoin script, even though it is very restricted, still can express a number of things that are very useful and interesting. But before that, let us see what the bitcoin script execution looks like. So let us say I have this after

concatenating, the sigscript. And then there is a script pub key, I have this so this part is actually part of the scriptsig.

And this is the script pub key and we have concatenated them so this is my complete script and I want to execute this in a stack based machine. So what happens is that the, this now one things that are expressed with the angular brackets are data because this is just an just a prototype example. So we did not have real data here in the actual case, there will be a signature there will be a public key which will be actually data.

And therefore, and this will also be data and register instructions. So, whenever we are seeing these angular brackets, those are data. So as I said that when I encountered data while executing, I push them on top of a stack. So let us look at the stack. So when I encountered stack, I will put that on the stack. Now next, I see that there is another piece of data so I have to put that on top of that. So I will put that also on top of the stack, the stack grows.

Now I encountered an instruction op_dup, so when I see op_dup, what it does is duplicates the top of the stack, so when it duplicates the top of the stack without removing what was on top of the stack, some instructions remove the entity on top of the stack, but the op_dup basically keeps the entity on top of the stack. So it creates another copy of this key and we will understand very soon why it did that.

Now, the next one is another instruction this instruction is saying that hash whatever is on the top of the stack. So, it is going to hash this top of the stack. So, and this one replaces what was on top of the stack. So now we have the hash of the public key on the top of the stack. Now, I encountered another data. So I put it on top of the stack. So I have now this data and this data. Then I have another instruction which says equivalenceverify.

So which means that take the top 2 elements of the stack this instruction semantics is that you take the 2 elements on the top of the stack and verify if they are equal. If they are equal, then you will it will return true, which means those 2 things will disappear from the stack, but nothing will be pushed on the stack. So, now, I have done that. So now I have only the public key and the signature.

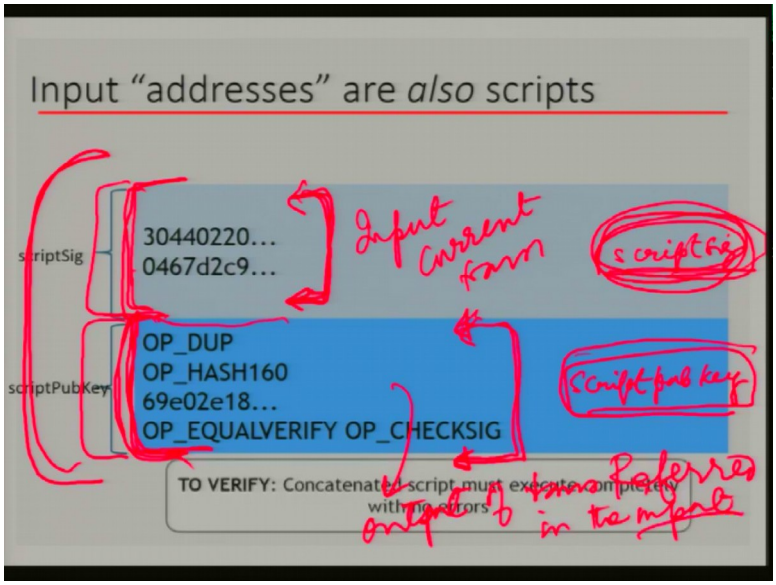
And then I encountered the next instruction which says, check the signature. The signature is done with the private key so if this public key is the correct public key for this private key, then we will be getting a verification otherwise the verification will fail. So if assuming that the verification passes, then it is going to be empty stack or it will be all true. And then I have succeeded in executing this entire script.

Now, let us have a little look at this a little bit more closely. So the guy who is spending the money he does a signature and he gives his public key because without the public key one cannot change, check the signature. So he gives his public key. Now if the users want, they can check from other places, because public key is a public information that this is indeed his public key. So he puts the public key, and he signs with the private key.

So this is the input part of the transaction gives the output part of the transaction is supposed to only give you the public key hash, of the person who is supposed to get the money. What it does is that it I am going to check whether this public key hash matches with the public key of the recipient. So for this, what I am going to do is that I am going to duplicate the public key, and then I will make a hash and I am going to compare it.

And then I am going to check verification of the equality and if they match then we know that this is the correct hash, and then I will check the signature, I have the public key of the signature, so I can check the signature. So, this is how this thing actually make sure that the spender of the money is indeed the correct spender has the authority to spend that money.

(Refer Slide Time: 06:26)

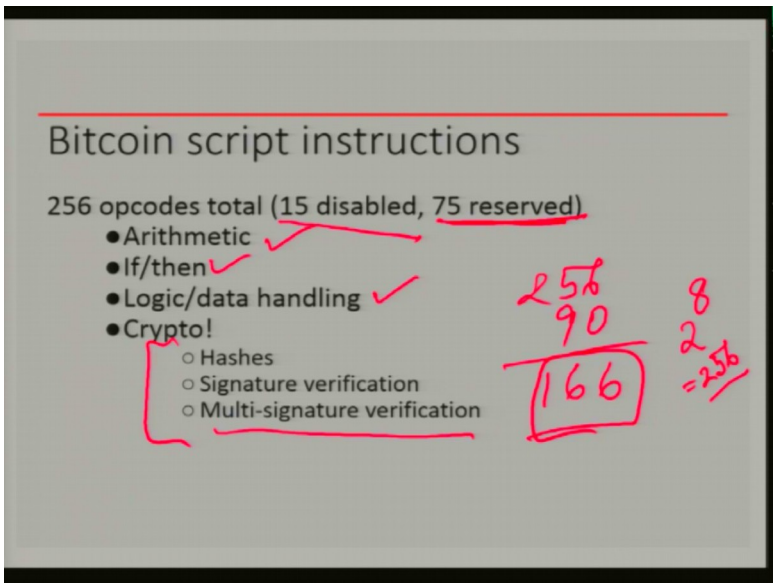


Now, remember, one thing I forgot to tell you is that you actually take the, this part so when you put together as shown in slides, you take scriptsig from your current transaction and scriptpubkey part you take from the previous transaction. Why? Because see, this is the part current transaction in the input part I put scriptsig I signed it and scriptpubkey is the in the output part of the previous transaction, which I am trying to use.

So this is not the output of the current transaction. This is the output of the previous transaction. And this is an the transaction which is being used for as input and scriptsig part is from the input so in the input I put my signature saying that I am referring to so answer transaction on which so and so output 0 or 1 or 2 whatever is I am going to redeem. So, what this has to do is that it has to take that corresponding outputs from the previous transaction corresponding outputs, output script pub key and then concatenate them together.

So, it is not that the input of this and the output of the current one our input and output of the current transaction are concatenated it is the input part of the current transaction and scriptpubkey part in this part is from the output of the transaction referred in the input. So, that is very important otherwise you will be confused when we when you try to understand this part. So, this is the how the execution happens. So it is completely stack based.

(Refer Slide Time: 08:03)

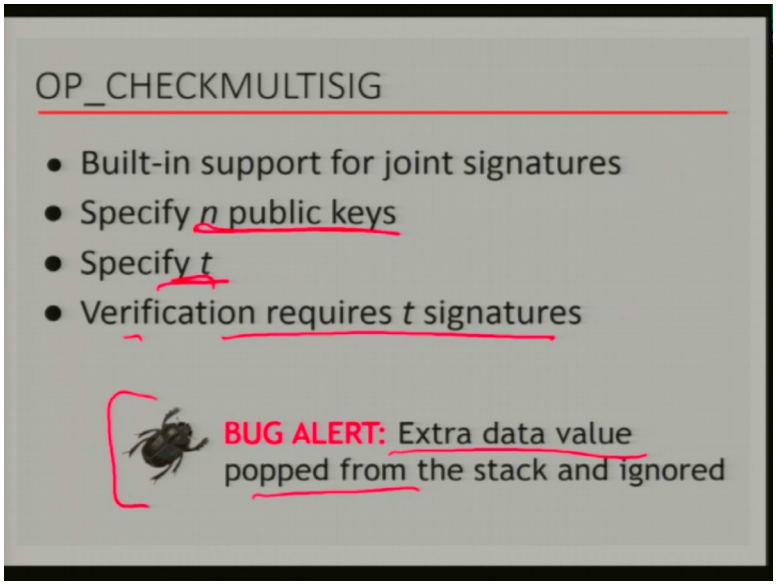


And the script bitcoin script instructions 8 bits long. And so there are 2^8 that is 256 different possible opcodes. But now there are 15 opcodes, they are not used, they are disabled, and 75 are reserved, but those opcodes have not given any functionality yet. So, out of 256, you

subtract 90, so you have 166, possible opcodes. So some of the opcodes are just arithmetic. In our example, it did not see any arithmetic.

Also some if then, some logical data handling and crypto functions like hash computing hash, verifying signature, and verifying multiple signature, things like that. So that is what the very, simple, and small language.

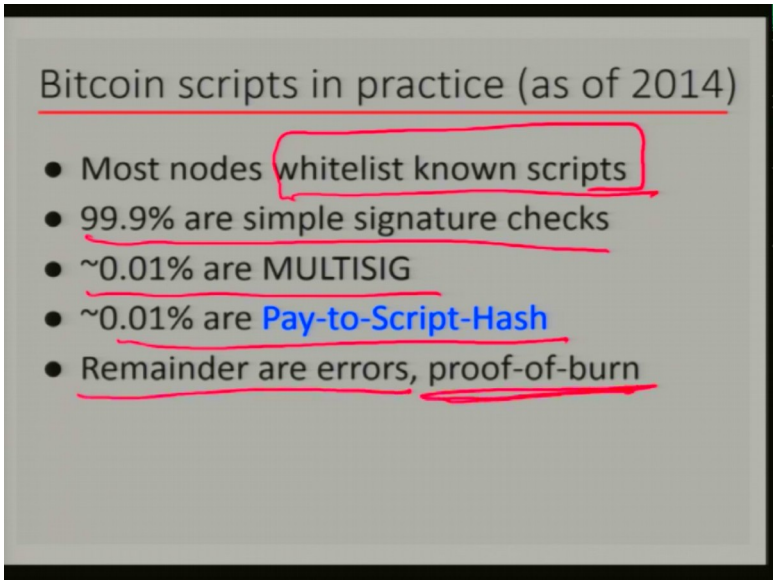
(Refer Slide Time: 08:55)



So multiSIG is an important opcode so, what it does is that it specifies n public keys and specify a number T and if you want to verify joint signatures, you record that t of them should verify. So, not all in necessarily have to be signing it to verify it but at least t of them should verify. Now if you specify T as N then you would require all the in entities to sign the transaction for the transaction to be valid.

Now, there is a bug which was there and in the beginning that there was an extra data value that is popped from the stack and it is ignored when the up op_checkmultiSIG is executed. Remember, each opcode when we when I get the opcode I often get the top of the stack or 2 top things on the stack are 3 top things on the stack and use it for executing the instruction. But in this case, they unnecessarily pop another extra value. And therefore, when you write your script, you have to add that repeat that data value that that gets unnecessarily popped, which is an annoyance, but it can be done.

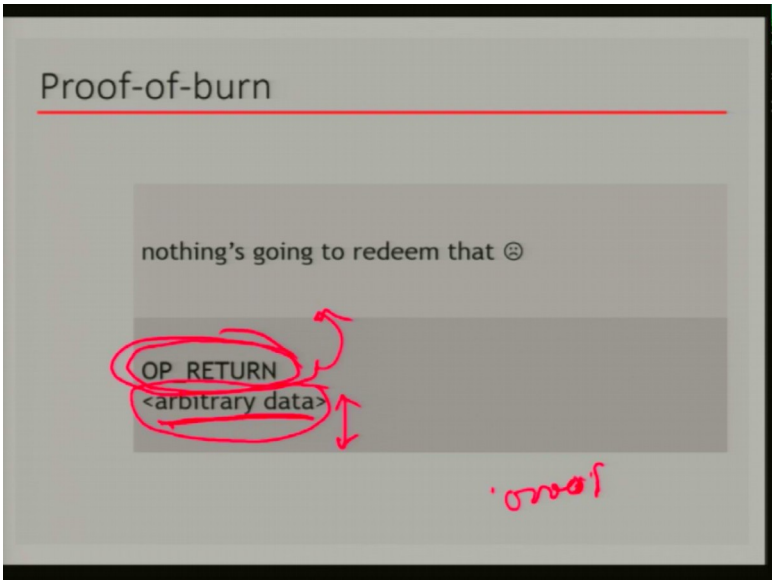
(Refer Slide Time: 10:22)



So in practice, most nodes whitelist known scripts with 160 or so opcodes, you can write many, many scripts, some of them actually was probably meaningless. But you can write a lot of scripts. Some of the scripts are maybe malicious doing something that you would not want to do and you did not know that it is malicious. So most nodes what they do is that they whitelist a number of scripts.

And if the script if any script defaults from their whitelisted set of scripts, then they actually did not even validate the transaction and declared the transaction to be invalid 99.9% scripts are actually simple signature check the one that we saw. And we executed on the slides. And about 1 in 100 is MULTISIG script. And there is another thing called pay to script hash that we will discuss shortly. That is another part another type of script that we see about 1 in100. And then remainders or errors and something called proof of burn, so we will also see what that means.

(Refer Slide Time: 11:31)



So proof of burn is an instruction. So what is a burn? Burn means that you are burning money. So if you take some bitcoin as inputs, and then you say that, but you did not specify who gets it, but the script executes fine, then that bitcoin becomes irredeemable which means that you cannot redeem it ever again. Because once you have used it as input, you cannot use it as input to another currency.

On the other hand in this transaction, you did not create an output. So, you cannot refer to that output to do the next transaction. So that that bitcoin goes, but you know nowhere so, nobody can use it. So, we say that we have burned that that amount of bitcoin. So, why would I need to do that? The reason is that the way the proof of burn happens is that you there is an opcode called op_return which returns without error.

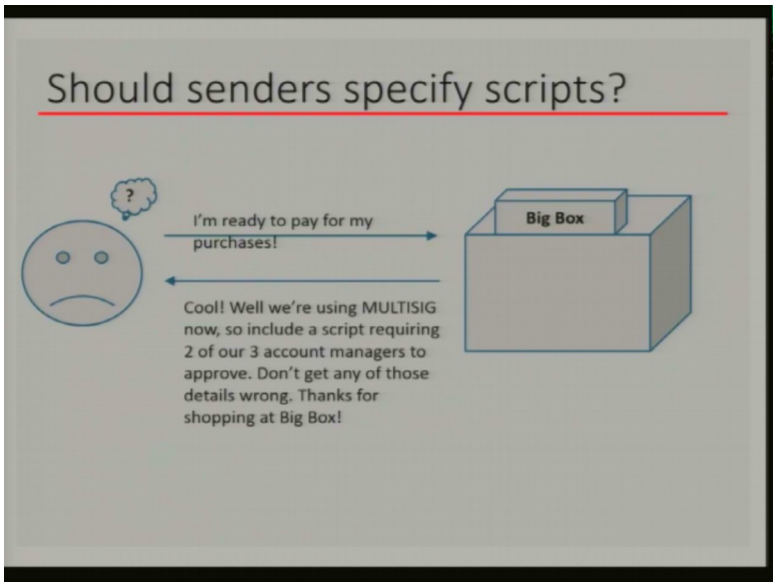
Now, we said that if when I execute the script on the stack based machine, then if I have no errors, then we assume that the transaction is valid. So, in this case, the input goes away, but the output is not gone is not going to any anybody. So, therefore, the money is gone. But what happens is that op_return is such that anything written after the op_return is ignored, because op_return basically returns the script from execution.

So, you can put arbitrary data after the operator and then the transaction will be valid and then this transaction will be eventually go into a block and will become permanent in the blockchain. So, if you want to keep some data forever into the blockchain then you use a proof of burn you pay a very small amount of bitcoin maybe 0.00001 amount of bitcoin burn it and then basically etched the data forever in one of the blocks.

So, many times you will hear that people are nowadays giving like your course certificate on the on bitcoin blockchain, this is how this is done. This data about your certificate will be actually etched into a transaction using a proof of burn transaction and then it will be in some block and so, if somebody wants to in future wants to check if you have the certificate, they can actually go into the bitcoin blockchain and retrieve that information.

And since this is supposed to be non temporal and supposed to be permanent, then this is a proof that you really got the certificate and you did not forge the certificate. So that is how this is done.

(Refer Slide Time: 14:17)



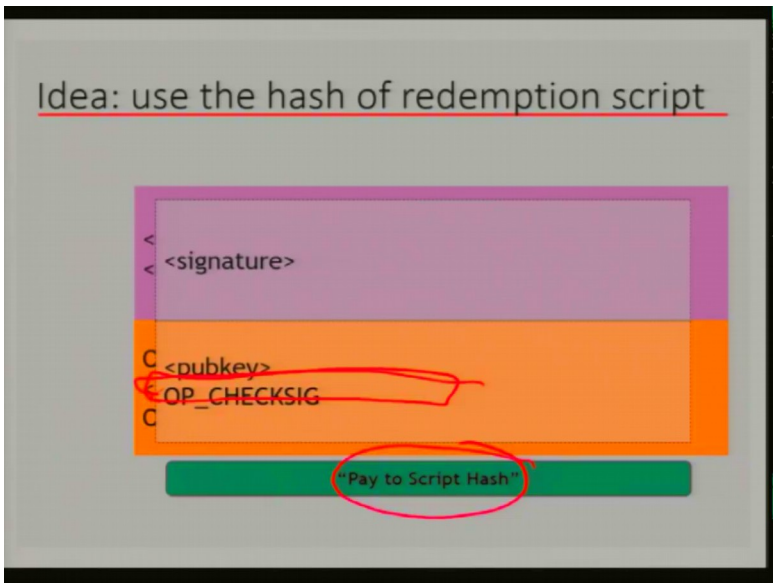
So, there is another issue that we saw that the script that we saw as the scriptpubkey was pretty long and not very long, but it was long and it has multiple instructions and data values in the places and so on. Now, suppose you are working with the merchant and you buy stuff from him. And then one day he says that look, we have changed ownership and we now have multiple owners.

Now we actually need to sign 3 of us need to sign before so you have to send the money to a multi SIG address. Now you get confused because you are the sender. And you are used to just regular writing scripts, which has basically looks like what we just saw is the checkSIG, type of thing. When it is multi SIG, you did not know. So you will be confused. And if you make a mistake in writing the script, then that is the money cannot be redeemed by the merchant.

So what the merchant can do is that he can give you a script. But he the way he does it is that he writes a script and puts us puts a hash pointer to that script. So when you send the money, you did not have to write a long script, all you have to do is to actually send the money to that hash. So this is a technique that was introduced later in blockchain. It was not there in the beginning. And this is called a pay to script hash type of payments.

So instead of paying it to a regular address, person's address, you are sending it to a hash value, which looks like a regular address. So you from the sender's point of view, there is no change. But from the perspective of the other side, you actually have a much more complex script that you are hiding from the sender.

(Refer Slide Time: 16:17)



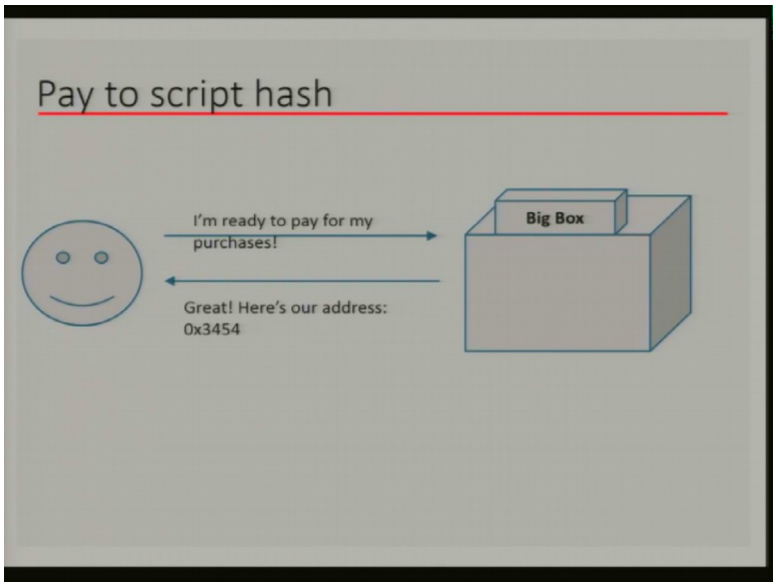
So what you do is that you put the hash of the redemption script, and then you check the redemption script. And if the hash matches, So, this part checks whether the redemption script to which you are sending is indeed the redemption script that you meant, this is all you have to write, like you did in case of a regular address. So, it could have been a regular address and you would have written the same thing.

You would have written the OP_hash and you would have written the hash of the redemption script and then equal and then so then you do the pubkey, then signature check and all that stuff, which was you would have done anyway. So, but what happens is that after you have done this, the actual script is replaced and that that script needs to be executed before the actual, the entire script finishes with, success.

So that is how, this works. And this is called a pay to script hash. So the basic idea is that you actually have to the receiver has a complex business situation where he needs the payment to be made to go through a complex script. But you did not want the burden of writing a complex script. So he makes it easier for you by writing a script and tell you the hash of the script.

And then you basically use the hash of the script as if you are sending to a regular address, which also is a hash of once, you know, size 160. So that is all there, to this pay to script hash.

(Refer Slide Time: 17:59)

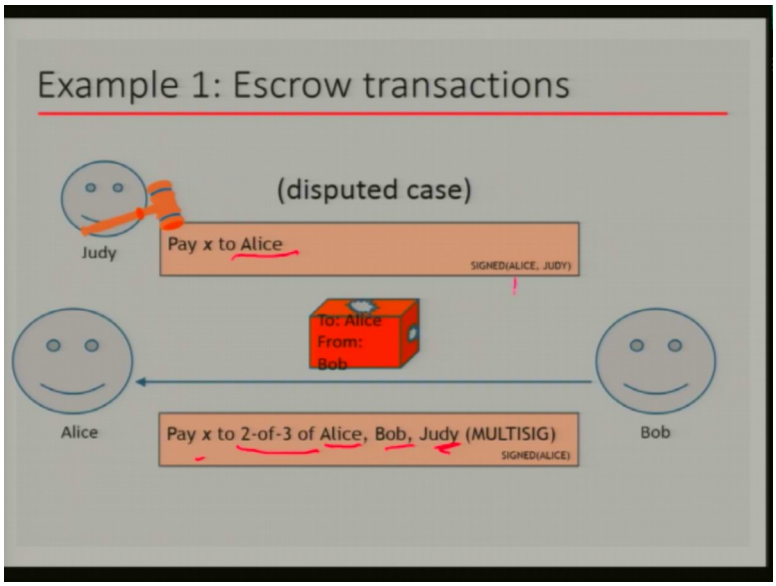


So that is how it works when you are ready to pay the purchase, these guys give you and basically a hash value as an address, you use that.

(Refer Slide Time: 18:07)

And that is what the pay to script hash means now, what we will see is that even though the language of bitcoin scripting, language is very simple, even with that, we can do a lot of smart things. And what we do with those is called Art sort of the precursor of smart contracts. So we can use scripts to do multiple things, where we will show you a couple of things.

(Refer Slide Time: 18:39)



So the first thing we want to show you is the escrow transaction. And then we will also see how to do micro payments, and then we will show you how to do a green payment. So these are the three examples will show but there are many others that you can do with this scripting language. But as we said that since this is not a turing complete language, we cannot do a very complex thing with this.

And therefore, we have a, we have to have a, you know, turing complete language if we want to do very, general things with the help of blockchain. And so let us see how what is an escrow transaction. So let us say Alice wants to buy something online from Bob. And Alice does not want to pay until Bob ships and shows the proof of shipment. But the bob does not want to ship until Alice pays.

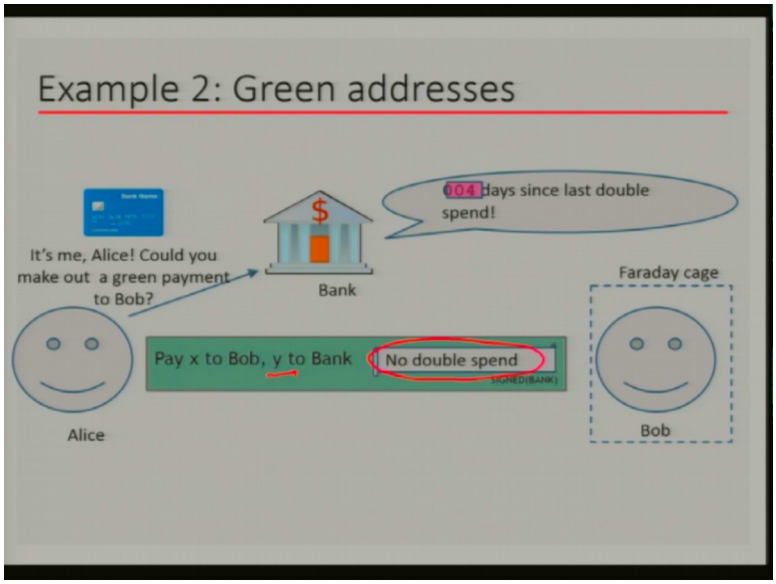
So normally, when we buy something from Amazon, what we do is that we paid by credit card, and then they actually ship the item. And then we can complain to the credit card company if the item is not good. And if we trust the merchant, then actually we can even complain to the merchant himself in like, like amazon, and maybe we can get a refund or something. But in this case, what escrow transactions do is that it allows you to involve a trusted third party to keep them money on escrow.

And then if both sides trust that third party, then the Bob the merchant would know that them even though he did not get the money, it is put in escrow and the escrowkeeper is trustworthy. So in case there is a problem, this crew keeper will pay the money to Bob. So what happens is that Alice writes a transaction that says that, that sends it to an escrow account, and it is it is a multi SIG instruction, which says that it requires 2 out of 3.

And Alice, Bob and Judy, who is the adjudicator or who is the with the escrow person, so now Bob knows that you have put your money in escrow so he ships the item to Alice. Now, in a normal case, suppose Alice gets the item and is happy and they are both honest. So then the, what the Alice says is that use this transaction of the escrow transaction as input and create a new transaction that is given to Bob. And it is signed by both Alice and Bob.

So that is that is the normal case. Now suppose what Alice gets is broken or has some problem, or it never reaches, then there is a dispute. And then Judy has to decide who is at fault. If, Judy decides that Bob is at fault, then she will sign this, and then we will get Alice will get the money.

(Refer Slide Time: 21:50)



If Alice is at fault, then Bob will get money. So this is what how the escrow transaction works. So you are basically putting the money in the in escrow until both sides are satisfied, it remains in escrow. If there is a no dispute, then normally, 2 sides sign, it is a 2 out of 3 signature. If there is a dispute, then Judy has to decide and Judy will sign with whichever party should get the money. Now, there is another thing called Green addresses.

So in green addresses, what happens is that say a similar problem Alice wants to pay Bob, but and Alice makes the payment transaction but Bob cannot wait for 6 verifications. Remember, like we said that there has to be 6 blocks built on top of the block that that contains this transaction in the blockchain in order to be you know, probabilistically more sure. So and also Bob does not want that Alice then goes around and does a double spending.

So all this stuff or Bob may be completely offline and he has no way to check what is happening. So what Alice can do he can go to third party. In this case, we call it a bank or it could be a crypto currency exchange, some entity, that what they do is that and they have a relation with Alice, the trust Alice or Alice might have a lot of money in the bank or a lot of crypto currency in the exchange or whatever.

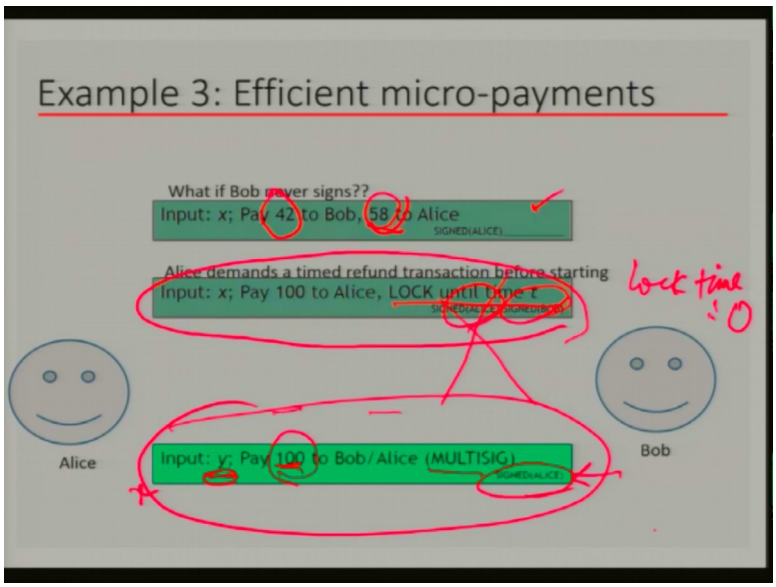
So Alice says, could you make out a green payment to Bob, and then the bank says, I will make a green payment to Bob. So what it does is puts a transaction that paid to Bob, and y to bank the white to bank parties for maybe for fees, that it charges Alice or whatever, and then Alice will then make a transaction with the bank to settle this. And then it gives a guarantee of no double spent, because Bob actually does not know Alice well, so Alice, he cannot trust.

And therefore, even if Alice paid X to Bob, and it comes on as a transaction on the network, Bob does not know that Alice will not now use the same input to another transaction and give it to her to give the money to herself. So that will be a double spin. Whereas bank is a more trusted entity so it probably would not do double spend because it worries about its reputation. Also, it may have some kind of a certificate that says that you know, it has not made a double spend transaction in so many years or whatever.

So Bob cross the bank and Bob then ships the item. Now, what happened in the double spend case is that is that the exchanges when they started this green address idea, people were excited because let us say you are ordering food, so the merchant cannot wait for 1 hour for 6 verifications to deliver your food. So green addresses idea was very welcome. So that the with the bitcoin transactions also you can buy things which can be delivered fast.

On the other hand, we merchant is also guaranteed to be paid eventually, even though he may not know the customer. But turns out that Mount Gox we talked about in a very beginning, Mount Gox was one of the exchanges that used to provide these green addresses in green payments. And there were other few other, but they all actually have run out of business. So green addresses nowadays is not very common because it did not really work out for other reasons, mostly outside the technology reason. But so green addresses in a good idea, but it did not really work that well.

(Refer Slide Time: 25:42)



The third thing that we want to talk about is efficient micro payments. So what is a micro payment say Bob is a mobile service provider and it charges Alice by the minute. So when Alice is making a call, then one possibility is that Alice is, whatever client she is running as the minutes take, she pays Bob, you know, payment for 1 minute and then and make another payment for a second minute and another payment for third minute.

But every time she does that, first of all, she will put too many transactions on the network, because this scenario could be very common if it is let us say airtel and alice is an airtel user and therefore this is not very efficient. Second thing is that the Alice might incur fees per transaction. So she does not want to do so many transactions. So, she has to pay more fees, it may be like you know, per minute payment may be less than the payment for the, for putting transactions into blocks.

So therefore, the idea of micro payments came into existence. So what we do is that Alice initially pays an amount y amount, let us say 100 are the maximum that Alice thinks that she would need to make this phone call from her whatever input transaction that she is trying to redeem. And then it is a multi SIG, and it is only signed by Alice. And this is not signed by Bob. So therefore, this is not going to be redeemed yet.

And then per minute, a new transaction is created, which is signed by Alice. Bob does not sign them. So now we are doing a number of double spend. So input y is your X transaction. And this is kind of like you made this payment. And then from this transaction, you are redeeming input x transaction. Although this transaction has not completed yet, you are redeeming this transaction with 1 to Bob and 99 to Alice.

And then next minute you pay 2 to Bob 92 Alice and so you are doing a lot of double spends. But none of them are being redeemed, they are actually not even put into the blockchain. Because all of this, this cannot go through because they are referring to the same transaction and doing various different things. And they are all double spends. But at the end when Alice says I am done, and Bob says, I will publish the transaction on the blockchain.

So what it does is that he signs the last one of them, and then publishes it. So what happens now is that the last amount that is due to Bob, so let us say it was 42 minutes. So then Bob gets 42, and 58 goes back to Alice. So all the other previous transactions are not put on the

blockchain. And therefore, and even if Bob tries to put them on the blockchain, it would not work because it will be a double spend. All these transactions are referring to this transaction.

That is why it is a double spent as we explained before, so now there is a possibility that Bob never signs this transaction. Now, what that means is that Alice will lose this 58 also, she Paid 100 before and then now, she has this 58 this one was actually put into the blockchain, but it was not you know signed by Bob, but now Alice has a possibility of losing the 58. So, therefore, there is a thing called timed refund transaction.

So, in a time refund transaction, Alice demand that when she does this one, Alice demands that the bob does another one in which Bob signs and says that pay 100 to Alice. And this is remember the lock time. The lock time is what we use here and therefore, the lock time will be set to some t appropriate t so that if Bob refuses to sign this one, then Alice can, bob already sign this so Alice can sign this and get all 100 back.

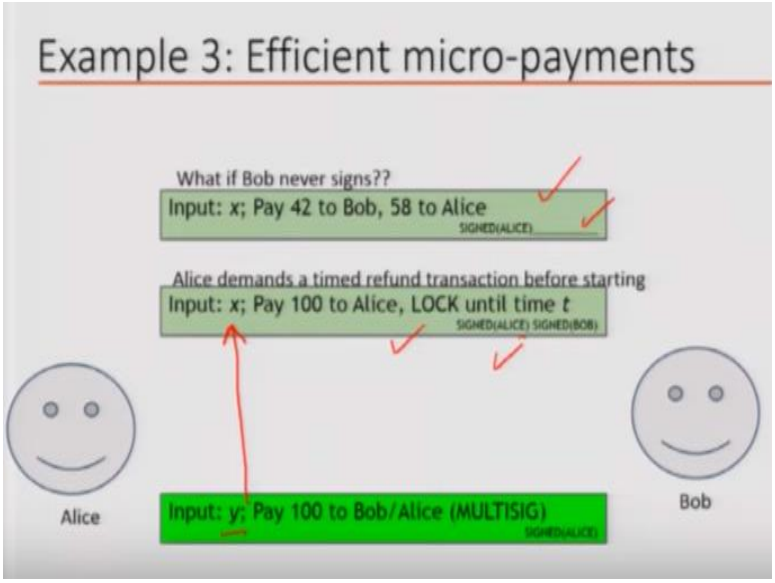
So, that is where the lock time thing that we saw before so usually we put lock time 0 for immediate use of the transaction, but here, we put a time t, and therefore, Alice can get back all the money after time t. We will come back to this in the next session. So for now, we have seen 3 different use of bitcoin scripts to do more complex transaction than just paying from one person to the other.

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

Lecture - 11

Welcome back. So we were discussing in the previous session about micropayments, and we explained to you how the micropayment works. Now we at the end we said that in case the other party does not sign the last transaction, which would have completed this micropayment into a single payment, then Alice is going to get locked out with some of her changes that she should get back.

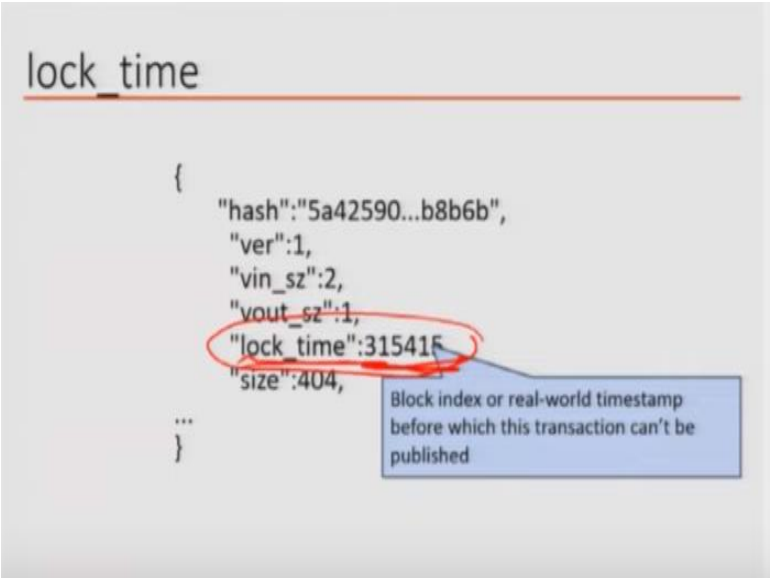
(Refer Slide Time: 00:46)



So therefore, in the beginning itself Alice takes in a demand from Bob a transaction that basically is a locked time transaction, which is signed by Bob and in which Alice is supposed to get back all the money. But it will be only be effective after time t. Now remember that this transaction is also using this as input. If this is ever accepted then this cannot be accepted, if this is accepted this cannot be accepted.

So once this transaction goes through it will not be effective. And if this transaction goes through this cannot be effective right. So that is the first thing. Second thing is that what lock until time t refers to is what we skipped earlier when we were discussing the metadata part of the transactions in bitcoin. So let us look at that again.

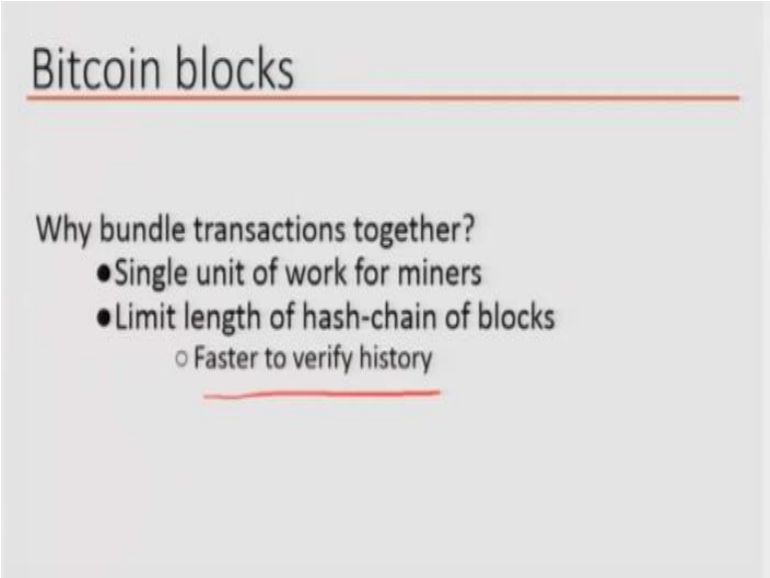
(Refer Slide Time: 01:37)



So this is the lock time feature and we earlier we saw it was zero which means that it is immediate transaction and here it says lock time as a time or the block number, right. So it could be the timestamp of the blocks. This transaction will not validate until that particular timestamp is arrived or crossed or if it is a block number then if that particular block number has been already is being mined.

So that is the idea of the lock time and this is being used in this micropayment transactions. So it could be a block index or the timestamp and this transaction cannot be published in the sense that it cannot be put in a block until that block number or the timestamp has arrived. So next so we have seen the transactions now we look at the bitcoin blocks.

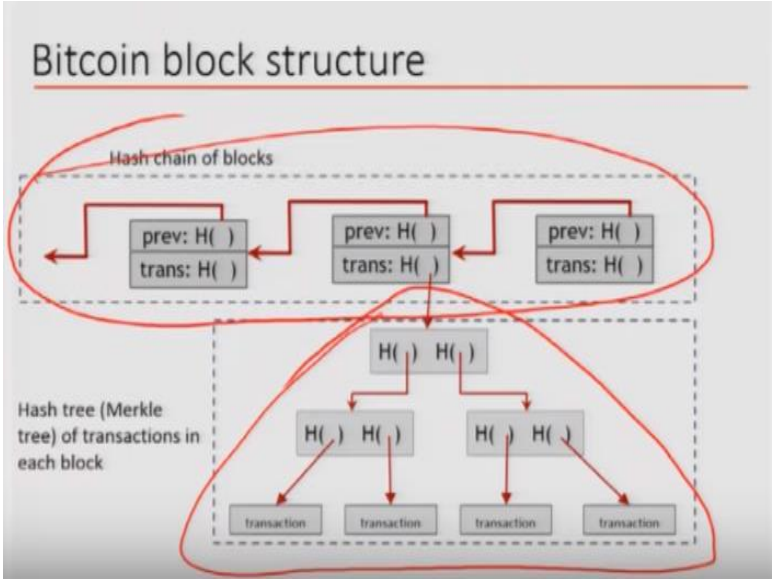
(Refer Slide Time: 02:36)



So blocks are when we bundle transactions together, we already know and then this is what the miners work on. So this is a single unit of work for miners and the length of the hash-chain. So if we had every transaction in one as a block, then you know you will have huge length blockchain. So therefore, when you put a lot of transactions together in a block, the number of blocks is much less.

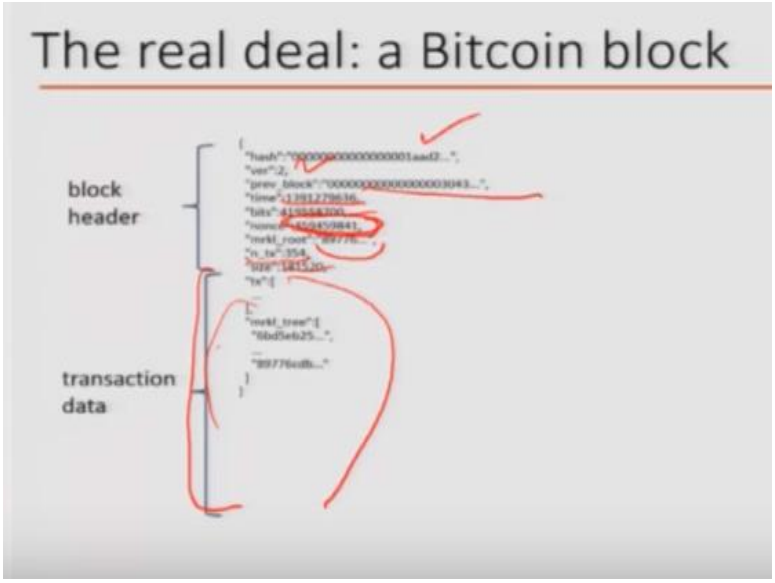
And therefore the length of the hash-chain is less. So that is the, and so if you want to verify history you do it block by block rather than transaction by transaction.

(Refer Slide Time: 03:12)



So the in the block structure, this is the actually the blockchain. So you have the hash of the previous block and the transactions and the transactions are put in a Merkle data structure. So this is how the transactions are stored so that you can quickly search through the transactions when you want to check, verify some history. So this is the block and this is the Merkle tree. So the Merkle tree root is stored here in the block.

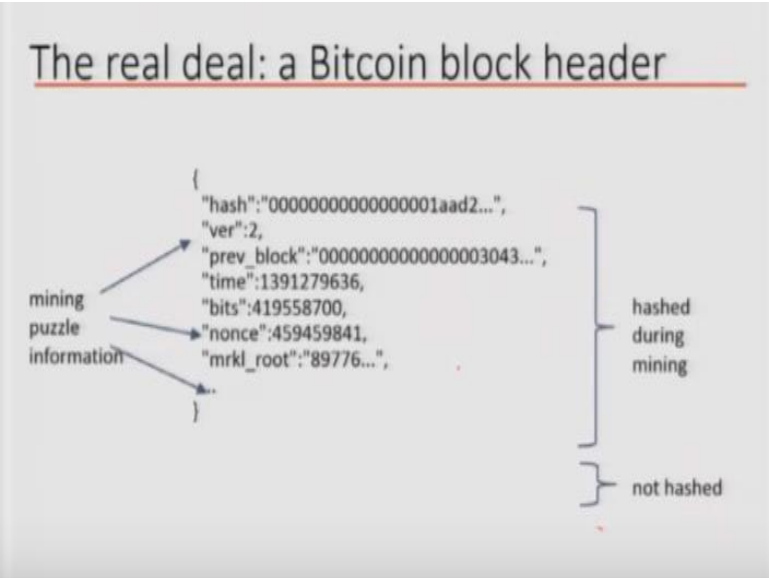
(Refer Slide Time: 03:39)



So the block header looks like this and this is where the transaction data is put. So this is the kind of like block header has a metadata. So this is the hash of the block. This is the version number. This is the previous block hash. And this is the timestamp. This is the number of bits. This is the nonce. This is what the when you solve the hash puzzle, remember there was a nonce that you have to discover in order to solve the hash puzzle.

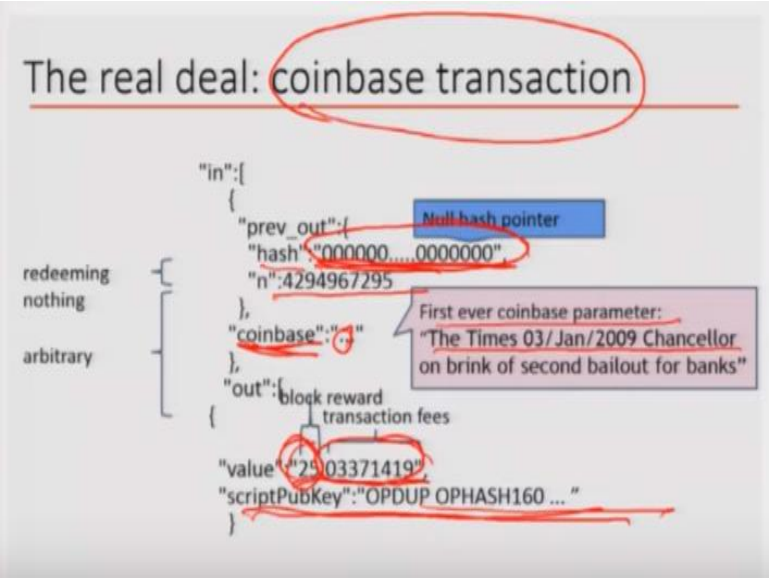
And that nonce can be checked to be the solution to the hash puzzle easily. So that nonce has to be stored here. And then the Merkle root, Merkle root is the root of this Merkle tree in which you have stood the transactions. And then you put the number of transactions, the size and then the transaction. And here is a Merkle tree for the transaction. So that is what the block structure looks like.

(Refer Slide Time: 04:36)



So this is what we talked about just now. These are all mining puzzle information, the version, the nonce, and some other maybe some other information. And then this is this part is hashed during mining. And the entire Merkle tree is not hashed. So this is the part of the block that is hashed for mining purposes.

(Refer Slide Time: 05:00)



Now there is when you make a block, you have to pay yourself whatever 25 coins or 12.5 coins whatever the current reward is. So anybody who mines the block, maybe his block will not eventually make it to the blockchain. But if I am mining a block and I have solved the hash puzzle, then what I do is I will create an extra transaction and put it into the block. And this is called a coinbase transaction.

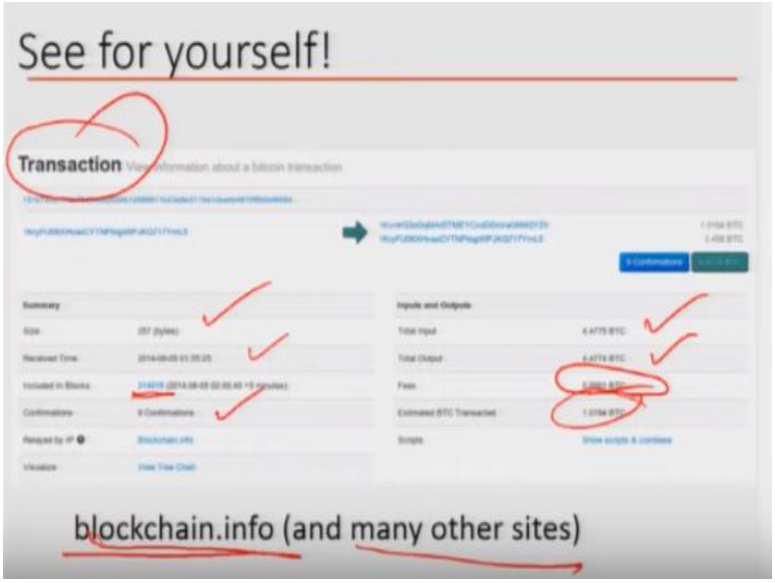
We mentioned this name before. So what the coinbase transaction does is that it creates a value of 25 or right now it is 12.5 and then add some transaction fee to it. And then you have this scriptPubKey that tells you that is one of your addresses, right because you are going to get the reward. So whoever is putting this transaction, see that the input is has is nothing. So it is a null pointer, right.

The hash of the previous output is all zeros, which means that there is no real input, right. And the output on the other hand is 25 point something. And right now it will be different, but we and then it is the hash of the public key of your address in which you want to get the reward. So this is what is called a coinbase transaction. In a coinbase transaction, there is also something here called coinbase.

And there you can put any information. So coinbase field can be an arbitrary information. And so this arbitrary information can also be used to store some information for the future. So what this is that in the first ever coinbase parameter in the very first block that is the genesis block the creator of bitcoin blockchain put this information. The Times 3rd January 2009 Chancellor on brink of second bailout for banks.

So we discussed this earlier that the bitcoin blockchain actually came as a rebellion against the 2008, 2009 slow down economic crisis, which was precipitated by the banking sector. So that seems to be this use of this particular arbitrary string in the first coinbase parameter kind of you know justifies that suspicion that that was the reason why the bitcoin blockchain was created.

(Refer Slide Time: 07:29)

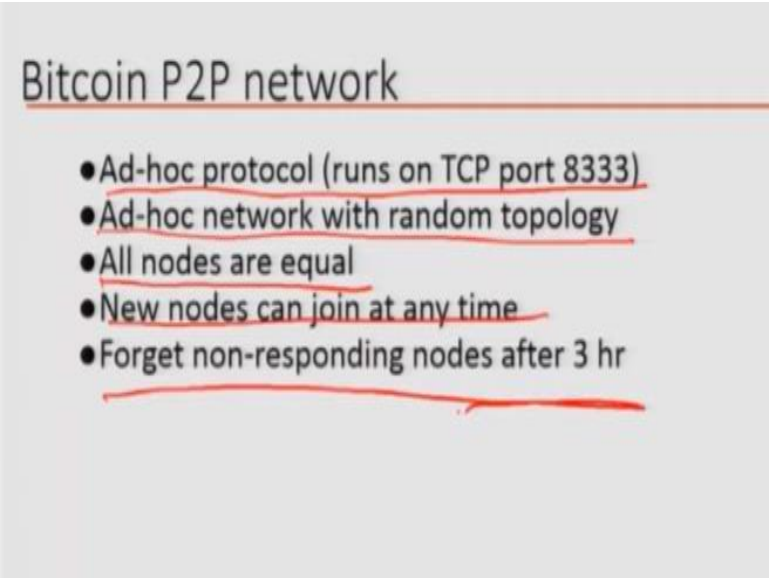


You can actually go to blockchain.info and there are many other sites. And you can actually look at browse through all transactions because the bitcoin blockchain is actually all public. So all the information, all the transactions all the blocks, they are all visible to anybody who wants to see and there are websites which get the data from the blockchain and make it viewable in nice format. So this is what you can see.

So this is about one particular transaction and it shows the size, the receive time, which block number it was added to, how many confirmations it has. And then you can actually see the tree chart. And also you can see inputs, amounts, output amounts. So there is a 0.0001 transaction fee given in this transaction and how many transactions, how many bitcoins were transacted, this kind of information.

So you can actually browse this kind of information by going to the blockchain.info to get yourself some familiarity with this. So now let us talk about the bitcoin network. So we said earlier, that bitcoin actually runs on a P2P network. The P2P network is basically an overlay network on top of the TCP IP network.

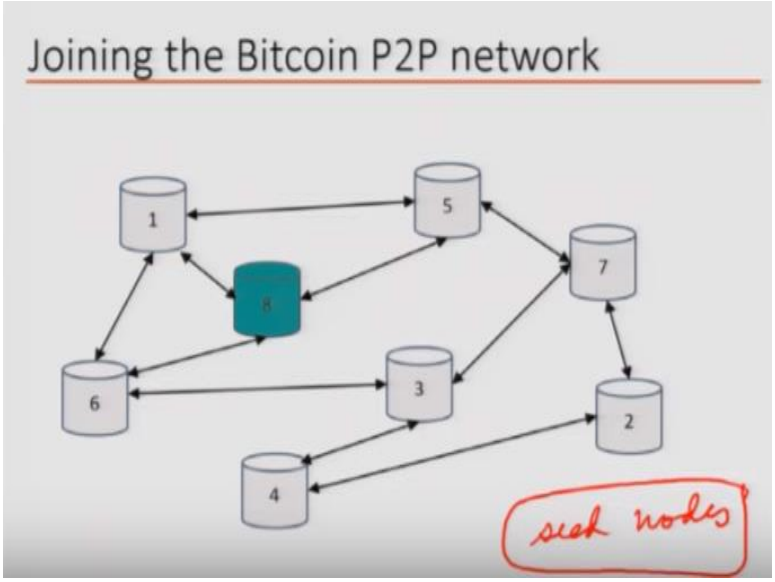
(Refer Slide Time: 08:51)



And on top of this P2P network is running the bitcoin application level protocol. So it is an ad-hoc protocol and it runs on TCP port number 8333. So an ad-hoc network with random topology gets generated as the overlay network on the standard internet. All nodes are equal. So there is no like a client server kind of architecture. New nodes can join any time, the barrier to entry is very small.

And also you can forget non-responding nodes after three hours, but the node can actually again wake up and join. So let us see how the joining works.

(Refer Slide Time: 09:32)



So the joining the bitcoin P2P network can be described with this figure, very nicely done by Arvind Narayanan. Here is let us say, the existing P2P network and suppose you want to join. So you say I want to join so you send out a message and there are

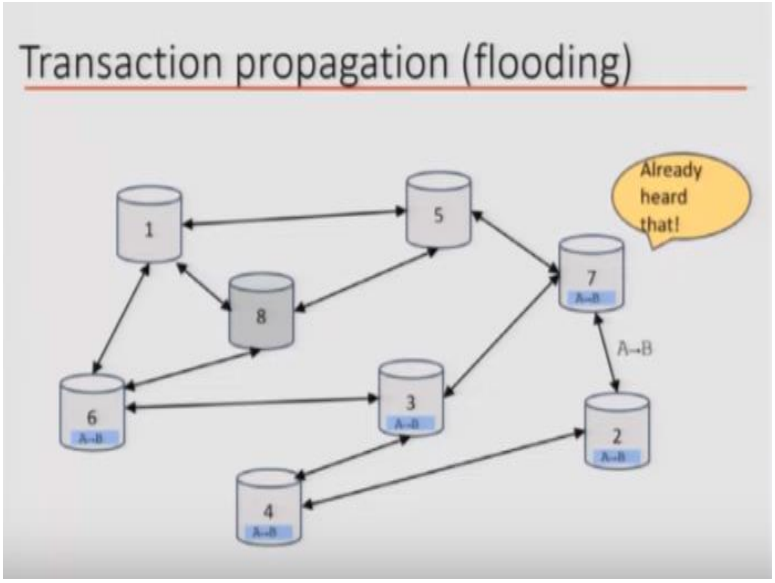
something called seed nodes. So seed nodes you can figure out most of the bitcoin software when you download them. It usually has the seed nodes listed in them.

So you can actually send message to the seed nodes first to start contacting the P2P network participants. So you let us say 5 is a seed node. So you send a message to 5 saying that tell me who else you know. Now 5 will tell that I know 1 and 7, I am connected to 1 and 7. So 8 will also know 1 and 7. So it will send message to 1 and 7 saying that who else do you know.

So 1 and 7 will then give him some more information, and let us say 1 will say I know 6. And then 7 will say I know 3 and 2, but 8 may choose since this does not have to be a fully connected network, 8 may choose that I will only be directly be connected to 1, 5 and 8, which means they are my neighbors. So whenever I want to forward a transaction or forward a block, I will only send to these three.

They can then forward it to the others they know. Or 8 may decide that I want to be more connected. So you might actually then get connected to 3, 7 and others. That is the ad-hoc nature of this P2P network.

(Refer Slide Time: 11:08)



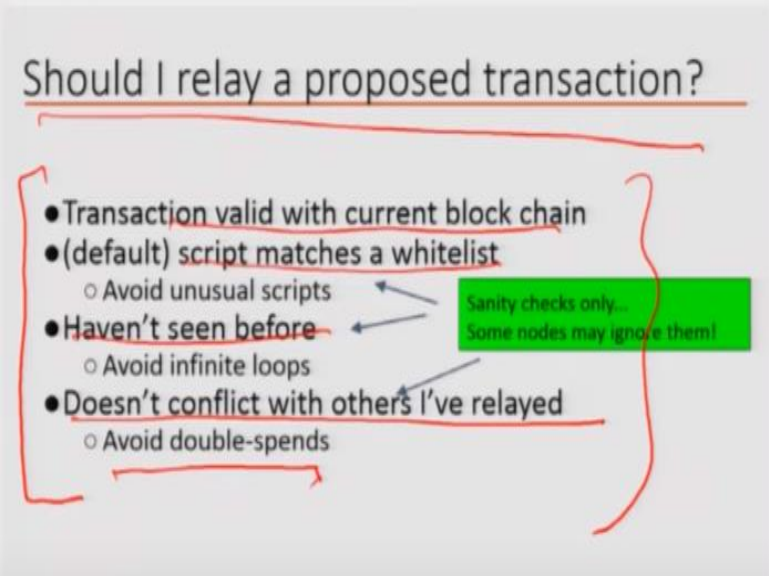
Now how do they do transaction propagation? So it is often called a gossip protocol. And also it is a flooding mechanism. So let us say there is a new transaction that 4 learns about or somebody who is actually connected to 4 puts that transaction on. So

the 4 will then send it to all its peer neighbors. So in this case, 4 is directly connected to only 3 and 2.

So it sends it to 3 and 2. 3 and 2 will check whether they have seen this transaction before. If they have not seen this transaction before, they will validate it. And if the validation goes through, then they will keep it in there you know pending transactions. The transactions that they are going to put in a block in case they are going to make a block. But they will also forward it to if they are satisfied that this is a valid transaction and this is forwardable, then they will forward to their neighbors.

So then 3 sends it to let us say 6 and 7. Now 6 and 7 now will learn about this. Now 2 will also send to 7. And then 7 will say I have seen this, so 7 will ignore it. So this is how this thing works.

(Refer Slide Time: 12:19)



So the question is, when do the nodes relay a proposed transaction? First of all, they check the validity of the transaction. Second thing they checks that the script matches a whitelist of scripts. So if the script in the transaction is unusual, they suspect that somebody is doing something malicious and they may not store it and they may not even forward it.

They must make sure that this is not an already seen transaction because if it is an already seen transaction, and if they still forward it, then there will be an infinite loop created. So anything that you have seen before you are not going to forward because

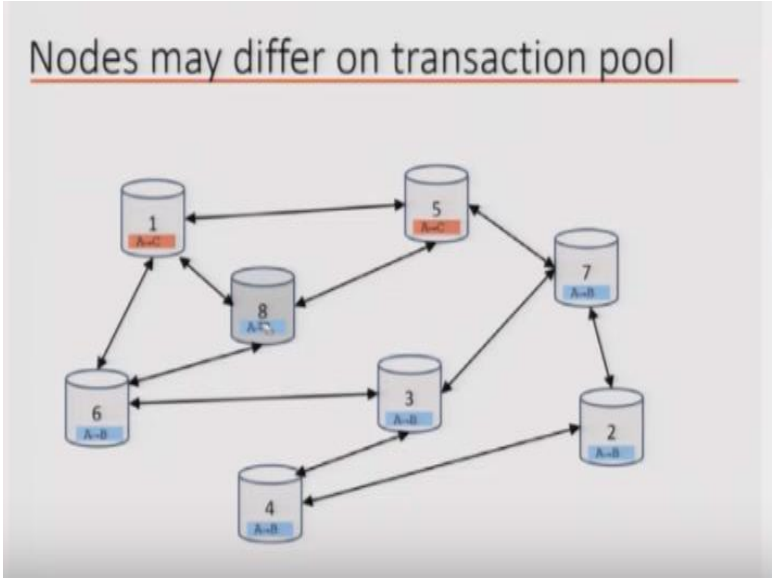
you have already forwarded it or you got it from somebody your neighbors. And it does not conflict with others that they have relayed. So this avoids the double-spend.

So these are the things that a good participant will check. Now if the participant is malicious, they may not check the validity and forward it. They might send something that they have seen before. They might actually not create whitelist and send any transactions out. So all that kind of stuff is also possible.

And therefore, the entire system should be designed in such a way so that even if some of the nodes does that kind of behavior, rogue behavior, not the prescribed behavior, then also the system works and it does not get into you know some kind of an infinite loop and other kinds of stuff. So we assume that most nodes will be sane and not rogue and therefore, they will stop.

Like even if you forward a malicious note, forward something but the others have seen it before they will not forward it. So at some point the forwarding will stop.

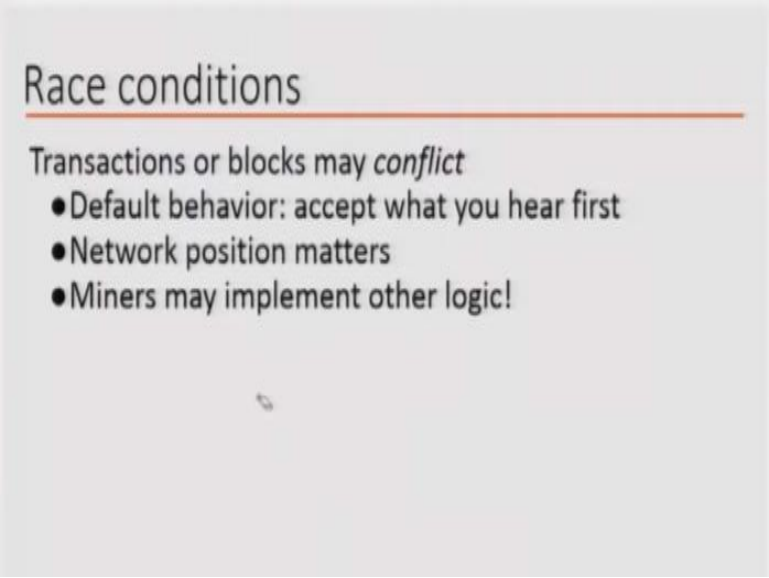
(Refer Slide Time: 14:03)



Now as we have seen in the previous sessions that nodes may differ on their pending transaction pool. So for example, you see that right now let us say these guys have these A to B transaction, and then one learns a new transaction A to C and starts flooding. So it sends A to C to its neighbors. 5 stores it but 8 may ignore it and 6 may ignore it. Now what that means is that there may be many reasons to ignore it.

Maybe 1 and 5 were malicious. So they are storing an invalid transaction, or maybe they are not malicious but they are the right ones and then 8 and 6 are maliciously ignoring this transaction. Anything can happen. And your system should be designed such that even if those things happen, because of the redundancy of mechanism in such a distributed system, eventually everything will work. That is the idea.

(Refer Slide Time: 15:04)

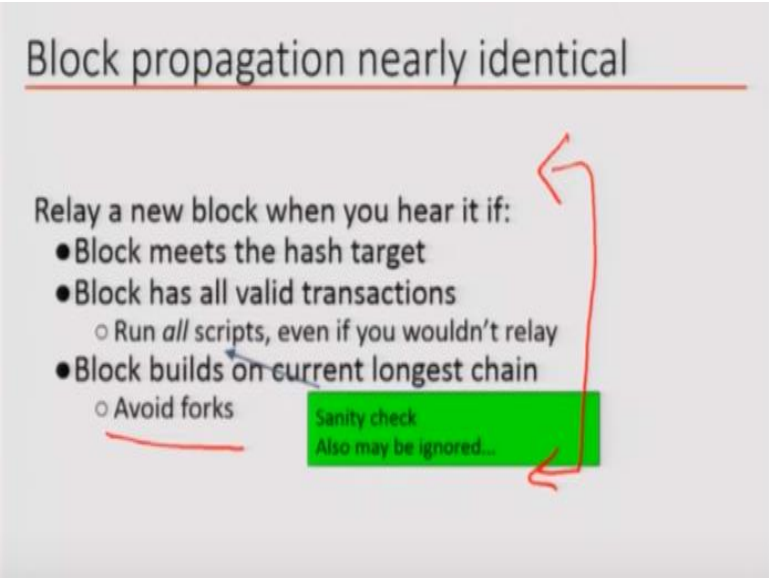


So transaction and blocks may also conflict, right. So as we discussed before that the nodes are geographically distributed very far, there is network delay and various kinds of things. So if I am close to the originator of the transaction or the originator of a block that is being broadcast, I might hear it first. But very far away node may hear it much later.

And therefore the blocks that I create would be different from each other. So each of the participant may have a different set of pending transactions and different set of blocks. So what they add in the blockchain or what they add in a block will differ from each other. So there would be some risk conditions that has to be automatically resolved at least probabilistically.

So with high probability, eventually the consensus branch will be unique in all the different nodes. So that is the goal.

(Refer Slide Time: 16:07)

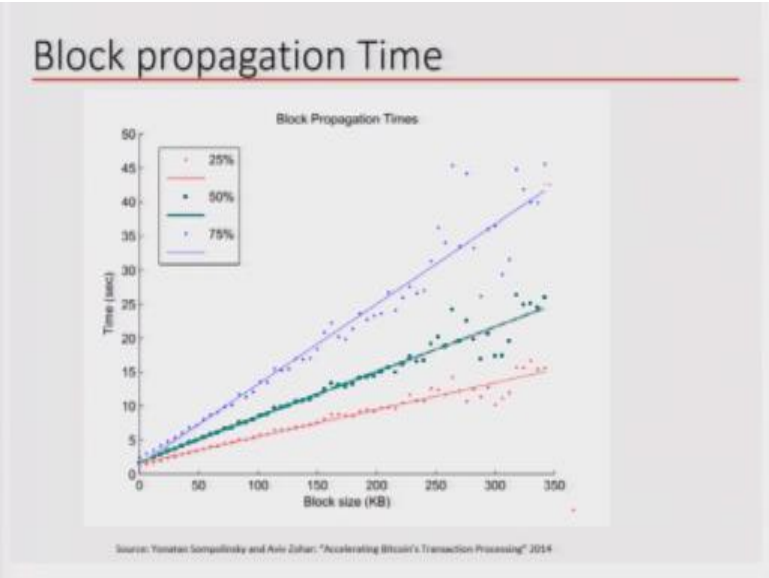


So the block propagation logic is also almost the same. You relay a new block when you hear it. First of all you have to check if the hash puzzle has been solved by this block by checking the nonce against the rest of the block. All the transactions in the block should be valid. So you check the validity of each transaction before you actually forward the blocks.

And the block is the, the current block in its previous hash is pointing to the last block in the longest chain. That because there may be some sub chains that are being created temporarily and then they get orphaned. But right now whatever is the longest chain, you should be building on top of that. So and this is to avoid forks. Again, this is what they say nodes or good nodes will do.

Now a malicious node can differ and do different things. And the protocol should be eventually robust to some of the nodes behaving differently than this.

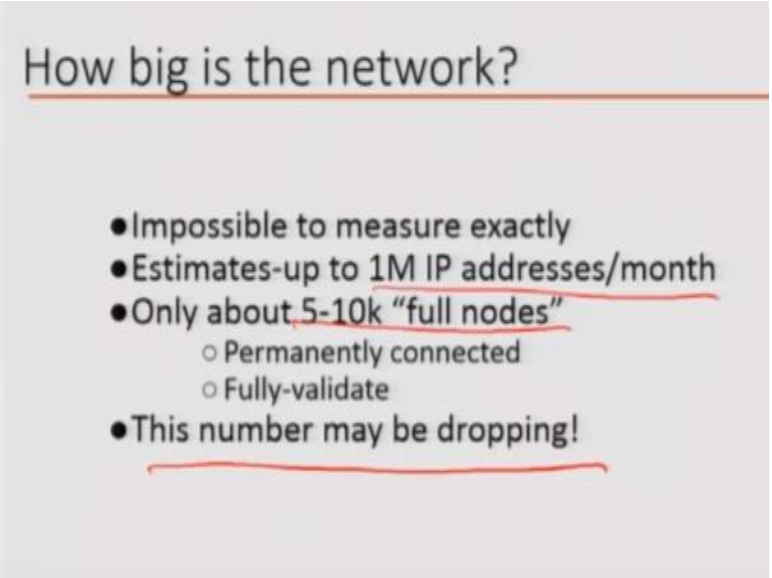
(Refer Slide Time: 17:11)



Now if we look at the real block propagation time, so this is the quartiles being plotted. This is the 25% quartile, and this is the 50% quartile and this is the 75% quartile. You see that 25% of them reaches all the nodes in like 15 seconds. The 50% of them reaches in about 25 seconds. And then there are some which actually reach, takes about 40-45 seconds. So that is the overall propagation time.

And as the block size increases, the propagation time also increases. That is what we see here.

(Refer Slide Time: 17:52)

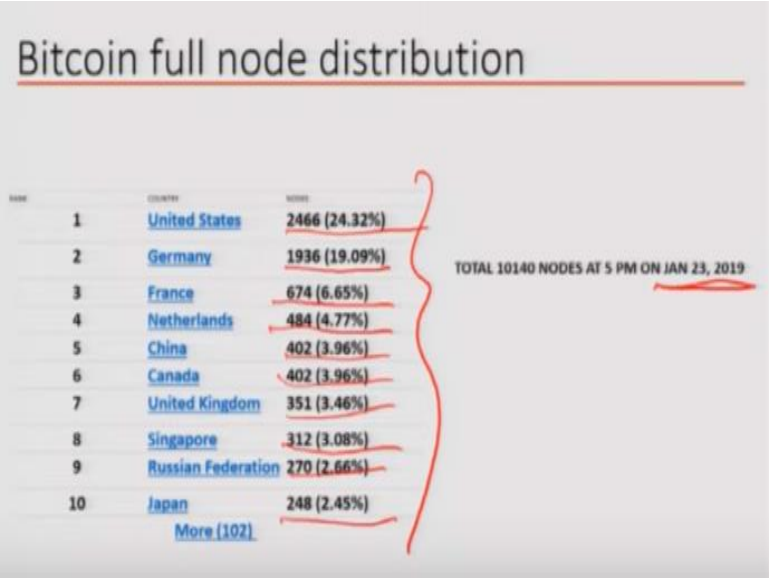


Now if you ask how big is this network, it is impossible to measure exactly. But it is estimated that up to 1 million IP addresses are involved in the blockchain per month, but only about 5 to 10,000 are full nodes. Full nodes are those which are permanently

connected, which have the complete blockchain copy. And they actually fully validate all the blocks.

They validate the hash puzzle has been solved, they validate that it is extending the longest chain and they also validate all the individual transactions in a block. And as the bitcoin is becoming less popular or less you know rewarding, this number may drop.

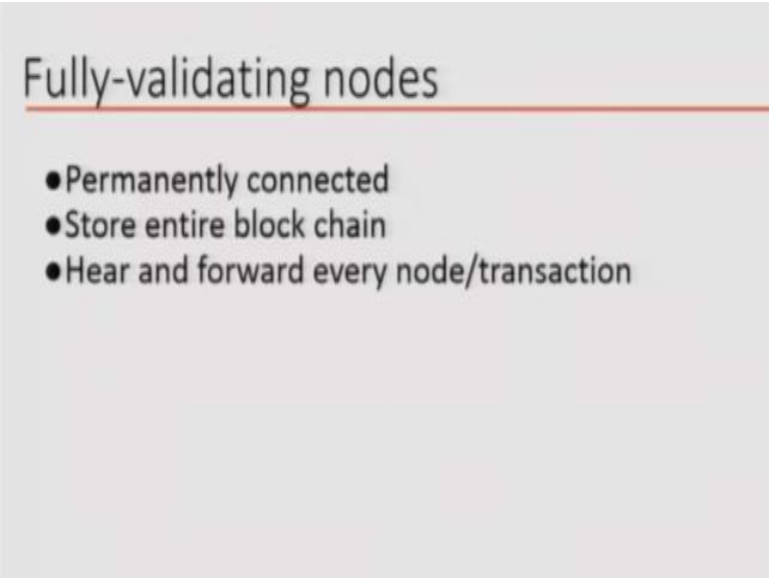
(Refer Slide Time: 18:34)



So let us look at the numbers as of January of 2019. So it is about 10,000 indeed at that time. And the United States had the highest number, almost 2500. Germany had almost 2000. France had about 674. Netherlands had about 500. China about 400. Canada about 400. UK about 350. Singapore about 300. Russia has about 270. Japan had 248 nodes. So these are the node distribution.

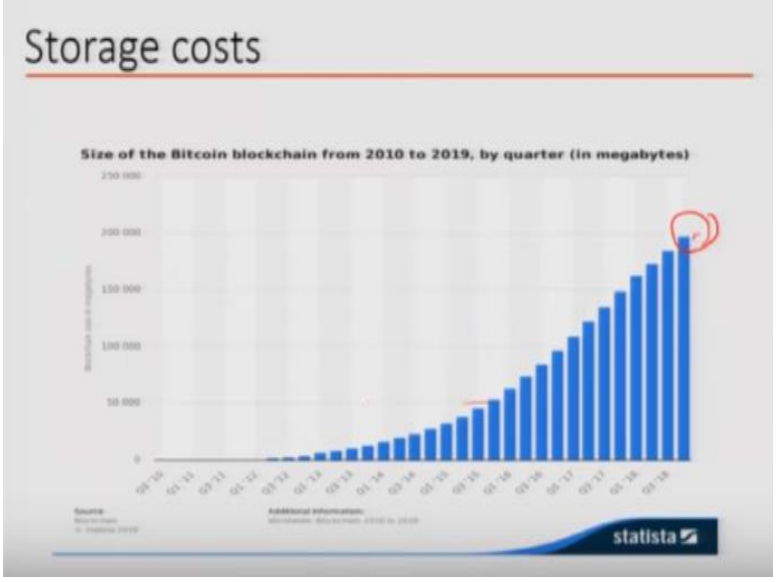
Full nodes require a lot of, although they may not be miners, but they are still going to be doing a lot of computation. And they have to be always connected. So that is a big deal to actually run full nodes.

(Refer Slide Time: 19:27)



So these are called fully validating nodes and they store the entire blockchain and they hear and forward every node and transaction.

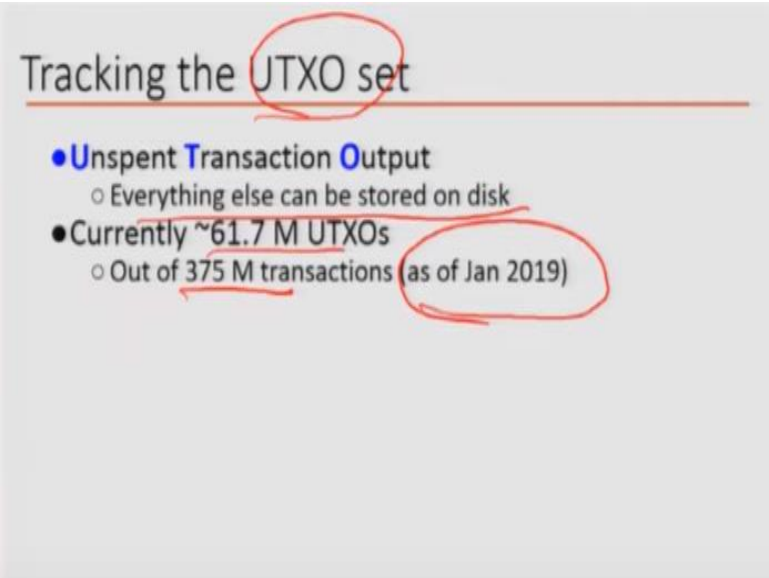
(Refer Slide Time: 19:37)



And then the storage cost. If you look at the numbers, right now it may be more but until 2019 it was about you know 200 gigabytes to store the entire blockchain. It may be about between 200 and 250 gigabytes. So you also have to if you have to store the entire blockchain, you need to have a lot of space also, right. 250 gigabytes is a lot of space. And it increased over time. It was in 2016, it was about 50 gigabytes.

And then by 2018, it was 200 gigabytes. So you can see how fast it was growing at that time.

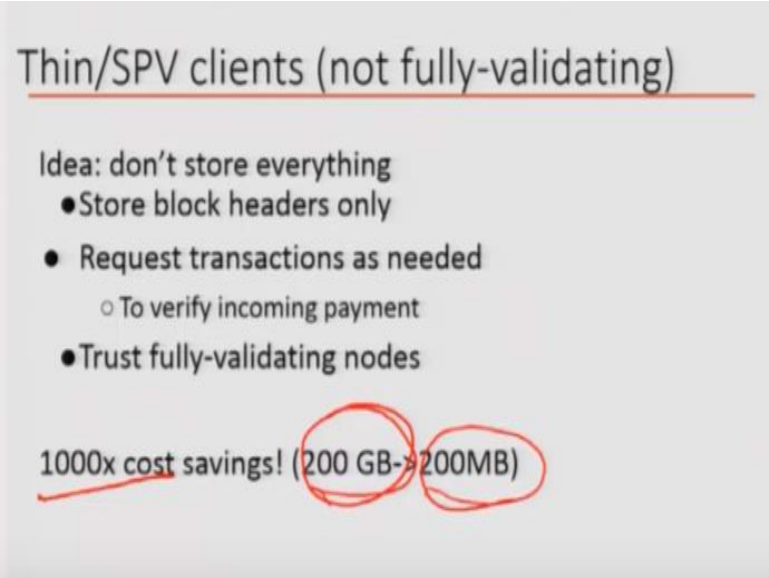
(Refer Slide Time: 20:15)



And also they have to track the unspent transaction output or UTXO set, because if a transaction is being referred to as input to a transaction, it must be one of the unspent transactions, everything else can be stored in the disk. So you can actually cache this information from the blockchain and rest you do not really need to use that often to validate transactions.

So at this point, as of January 19, there were about 62 million unspent transactions out of the 375 million transaction. So it was about one-sixth of the transactions. That is about 15% of the transactions were actually unspent. So you see that you still have to keep a lot of information around.

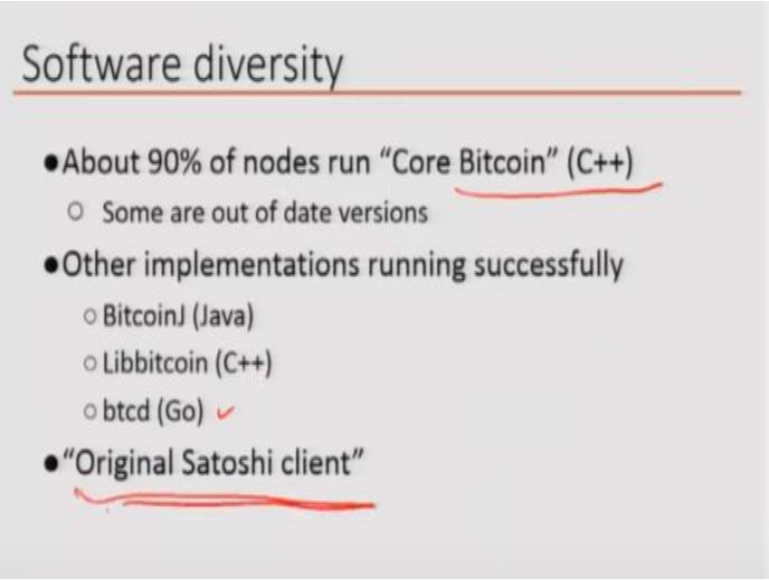
(Refer Slide Time: 21:05)



So thin clients which are not fully validating nodes, they do not store everything. They store the block headers only. They request transactions as needed. And they trust the fully validating nodes for doing the actual validation. Now these, just the headers the block headers will cost you at this point in January was about 200 megabytes. That is an actual full blockchain was 200 gigabytes.

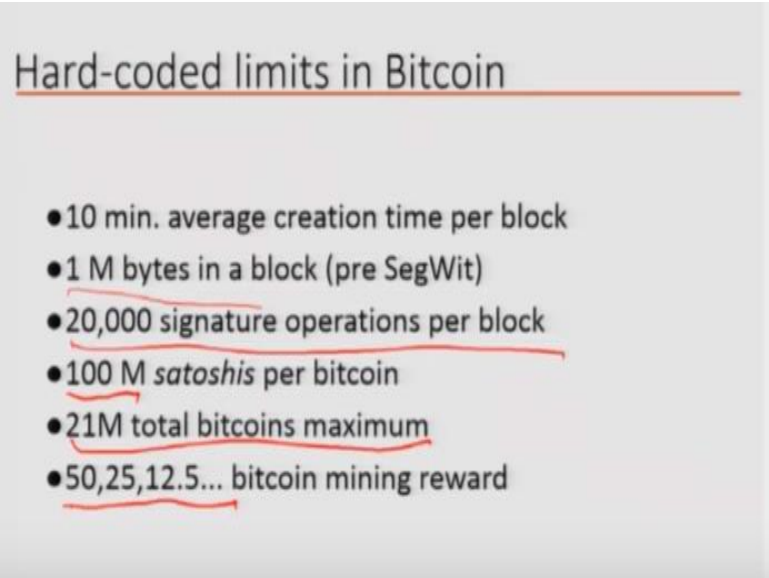
So it is about thousand times cost saving in terms of space. So wallets for example, if you have a blockchain wallet, that usually has an SPV built into it. It is a simple client. So they are not fully validating. They only worry about transactions that are relevant to them. So if somebody is paying your wallet money, then it actually keeps track of those and validate those.

(Refer Slide Time: 22:04)



Now there is a diversity of the software that it being used here. So for example 90% of the nodes run the core bitcoin that is C++. And then some there are some Java clients and there are some other C++ clients and there are Go clients. And then there is the original client that was designed by Satoshi Nakamoto. So we will now finish by discussing very quickly the limitations and improvements. So first of all, there are some hard coded limits in bitcoin.

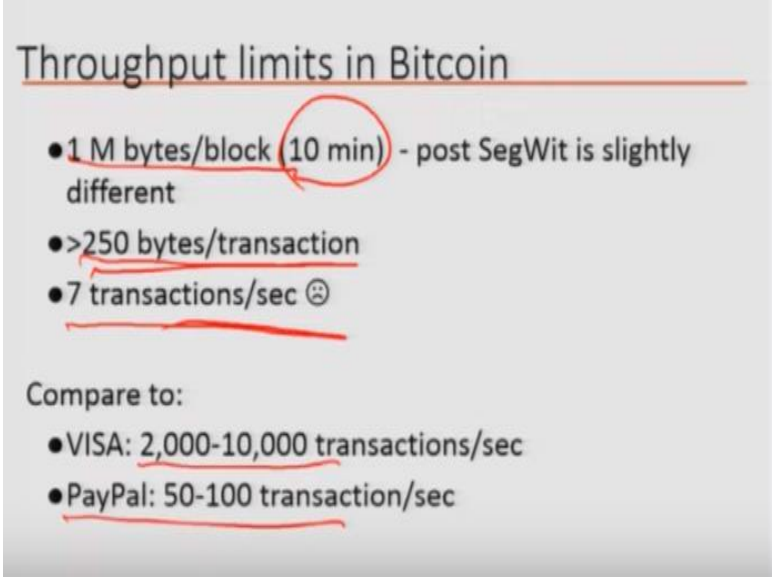
(Refer Slide Time: 22:35)



10 minute average creation time per block. 1 megabyte size for the block. There was a discussion on actually having a forking of the blockchain to actually increase it to 2 megabytes, but as far as I know it has not happened. 20,000 signature operations per block and there every bitcoin is divided into 10 to the 8 satoshis per bitcoin. So the resolution of paying bitcoin is about 10 to the 8.

And there is 21 million total bitcoins maximum, we discussed this before. And the only way to create bitcoin was through mining and the mining rewards have been decreasing and by 2040 it will reach 21 million when it will stop. So these are hard-coded limits of the bitcoin blockchain.

(Refer Slide Time: 23:29)

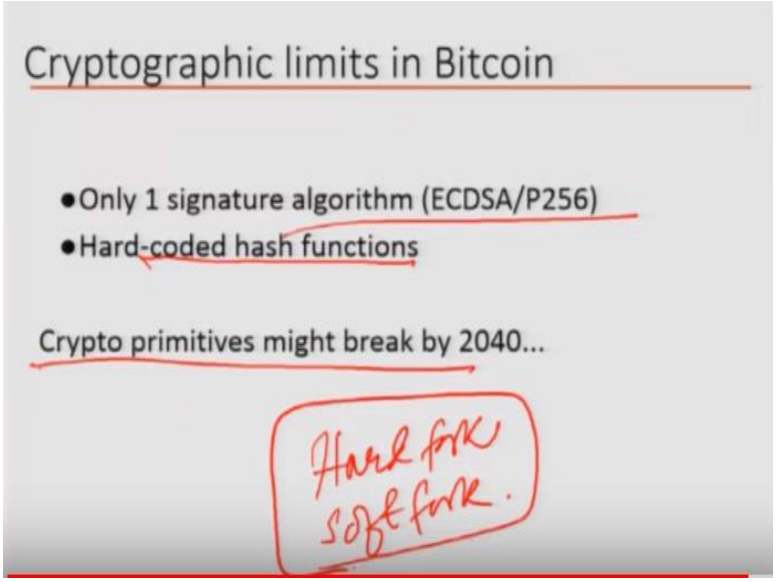


There is a throughput limit because if you have only 1 megabyte per block and each on an average each transaction is about 250 bytes, then you can have only so many about I think 40,000 or so transactions and they you know confirmed in 10 minutes because until the block is mined you have not seen it into permanence, although actually you have to wait for 60 minutes to see 6 confirmations.

But assuming that you do so many you are okay with zero confirmation even then you are actually having only a block full of transactions. And then that turns out to be about 7 transactions per second. And if you compare this to Visa, MasterCard, they can do 2000-10000 transactions per second and PayPal for example can do 50-100 transactions per second.

So compared to that, the throughput is very small, which basically is a problem in using bitcoin for real transactions.

(Refer Slide Time: 24:36)



So there is only 1 signature algorithm that is used. So and there is a hard-coded hash function that does it. And the other issue is that the crypto primitives that are implemented in the scripting language, they might break by 2040 because cryptoanalysis and cryptographers are always breaking the crypto's primitives. And then you have to redo the entire scripting language. The old scripts cannot be used again.

So there will be some problem and that will require something called a forking of the bitcoin blockchain. And so there are two types of forking, hard forking and the soft forking. And when we come back to the next session, we will talk about this two types of forking because this forking issue is true for other block chains also. So we have to understand what forking means.

So in this session, we will end here by just saying that there are certain limitations of the blockchain especially in terms of the longevity of the current version, because of the cryptographic primitives might be dated by 2040. And therefore will have some, we have to do some in the scripting language and that would require a change in the entire blockchain infrastructure.

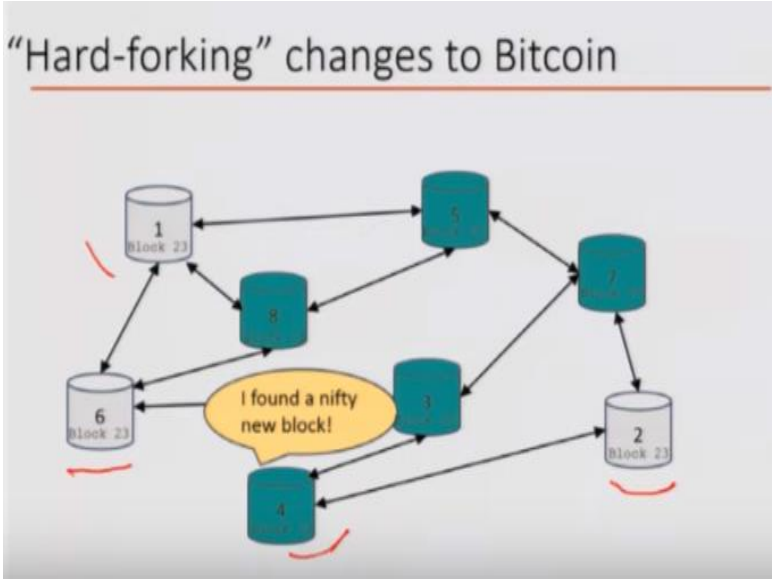
And that would require some forking and we will discuss forking in the next session.

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

Lecture - 12

Welcome back. So we have been talking about the bitcoin and it has certain limits, hardcoded limits and we were discussing that you know in the future, if we want to change these limits, what are the process or procedure for doing so.

(Refer Slide Time: 00:40)



So it turns out that since all the activities on bitcoin blockchain depends on the mining process, whether they will be accepted or rejected, and they will be made permanent it is not that easy to actually change the software or to change the limits, which are hardcoded or change the mining process or change the opcodes.

For example, in the scripting language, or to actually do a different kinds of hashing algorithm or signature algorithms, because it would require some work. So it turns out that what happens, so we already know that when a block is mined, the one solves a puzzle. But at the same time, multiple people, multiple nodes might actually solve the puzzle at the same time.

And everybody who solves a puzzle, which is computationally very expensive, they expect that their mined block will be made permanent in the blockchain, so they broadcast their block. Now depending on where you are located with respect to the

miners, your block versus somebody else's block might first make it to one of the miners. And that miner might actually attach that block to the blockchain.

Whereas some miner on the other side of the world, who gets another block, which also has solved the puzzle, and add that to as the next block to his copy of the blockchain, and that is what is called a forking. Now fortunately, over time, these forks usually converge. And so one of these two blocks will actually survive in the longest chain of the network. So this is not a deterministic guarantee.

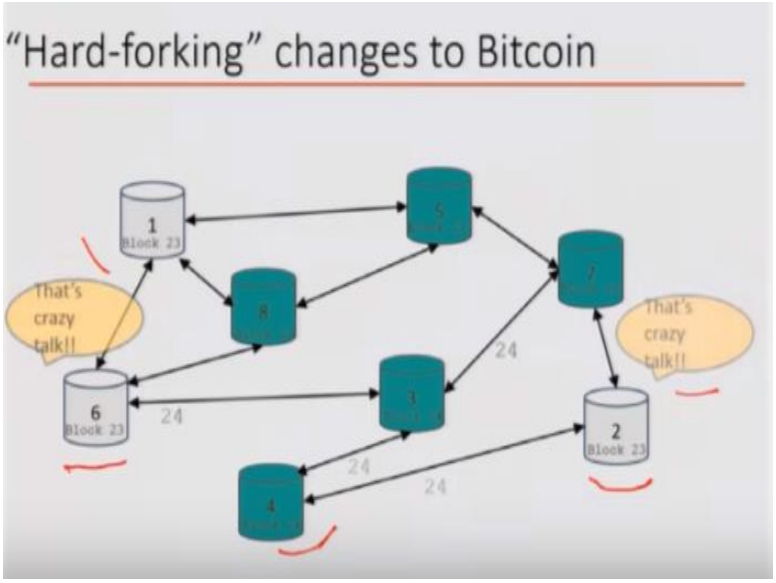
But this is a probabilistic guarantee with a high probability. So that is how a single chain which we call the consensus chain survives, and whatever is on the consensus chain is considered the permanent blockchain any transactions on any of the blocks on that consensus chain is considered permanent and but forking is a natural phenomena that happens.

However, when you want to change something intentionally in the blockchain code or in some of the processes, then there may be two possible things. One is called the hard-forking and another is called a soft-forking. So let us first look at hard-forking. So what happens is that suppose you make a change and some of the nodes upgrade their software to reflect that change.

So that change could be that you have changed opcode or you have changed the hashing algorithm of the bitcoin by you know re-implementing one of the opcodes. And then based on that, the blocks let us say, number 4 here, this node, mines a block and he expects that when he broadcasts this block, all the nodes will actually validate the block validate all the transactions inside.

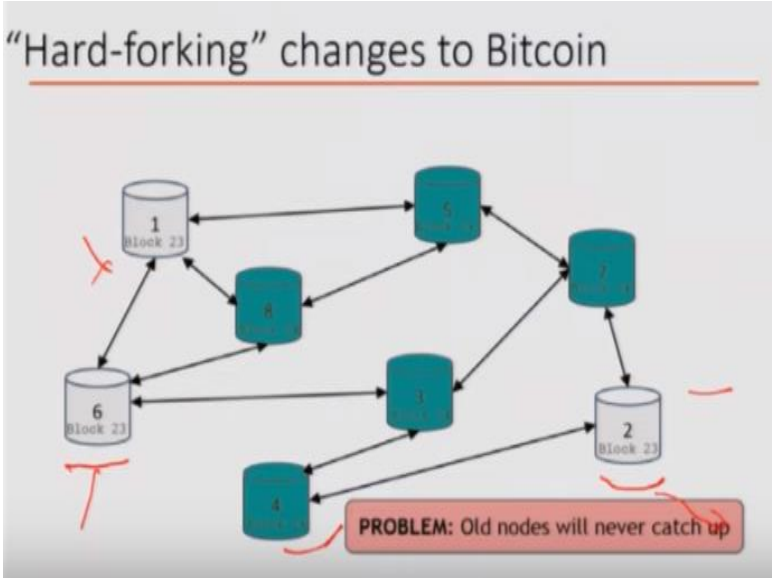
And validate that this solution to the puzzle is in fact a solution and then they will keep attaching it into their copy of the blockchain. But now here we have an example where some of the nodes did not upgrade and therefore, either intentionally or they are not aware of the change.

(Refer Slide Time: 04:32)



So when the block goes to them, they will consider that block to be invalid and they will not accept that block and therefore, they will not attach that to their copy. And this will happen with all the nodes that do not have the new upgrade.

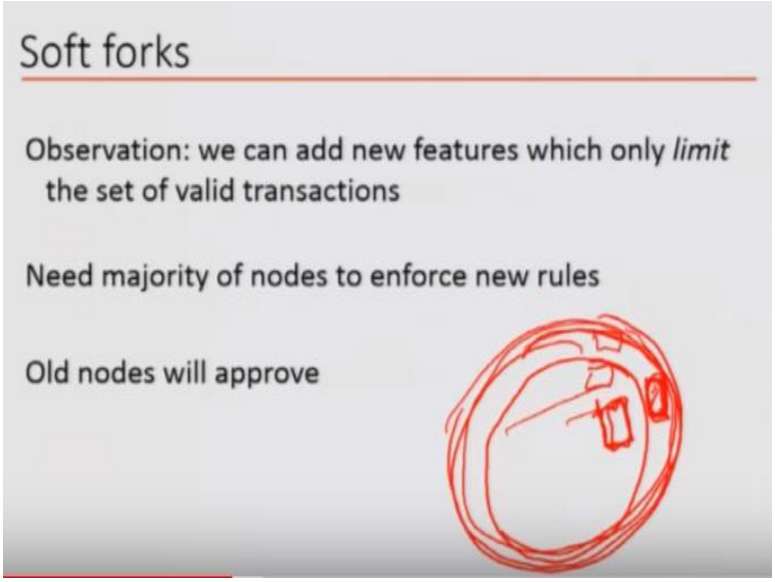
(Refer Slide Time: 04:50)



And this way, the copy of the blockchain that is being maintained by this nodes and what is being maintained by these green nodes will differ. And at this point, there will be two different consensus chains. And this will become suddenly two different blockchain with a shared history. So until this point when the upgrade happen the consensus chain versus a convergent chain, and then suddenly there will be two of them.

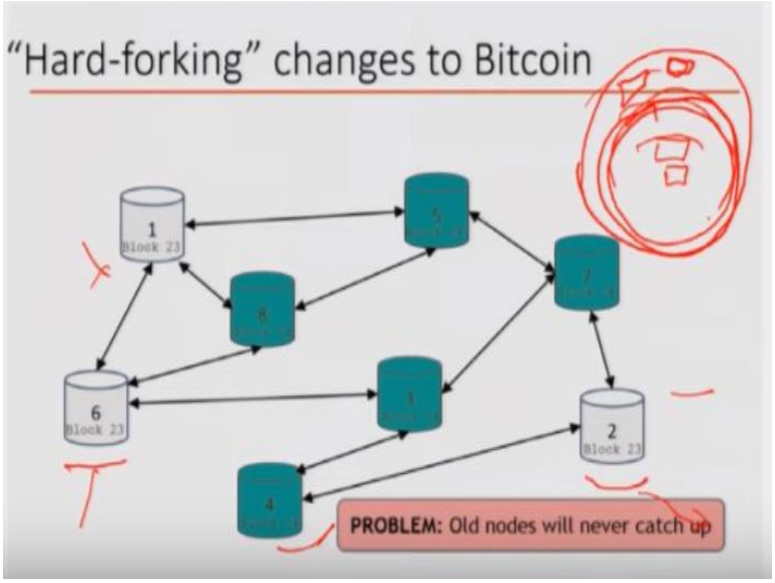
And that will require the blockchain to have two different things and accordingly the currency will also split. And this happened to bitcoin, like for example, bitcoin cash is one of the hard fork actually the original bitcoin consensus chain that continued and then the regular bitcoin is based on an upgraded version.

(Refer Slide Time: 05:45)



So one example and the soft fork would be when you add features, but so in this case, in the previous case, in the hard-forking case, what happens is that the let us say this is the set of valid blocks.

(Refer Slide Time: 05:54)



So now you are talking about set theory. So any anything that any property that is satisfied by these blocks can be thought of as a set. You know it may be a very large

actually maybe an infinite set of all possible blocks that are valid. And then you suddenly decide that you will increase the set of valid blocks, right.

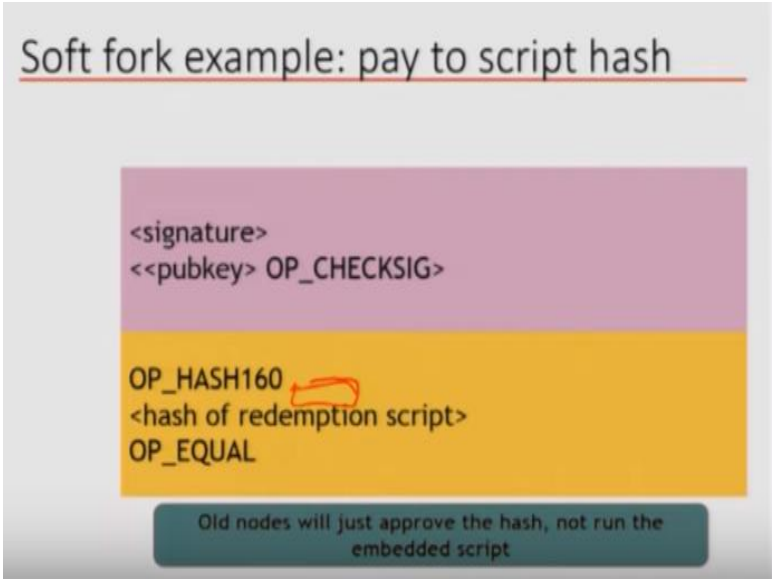
So now there are blocks here in your expanded set of valid blocks, that are not considered valid by this other nodes which are not aware of this extension or which do not agree on this extension. And therefore, the blocks mined according to the extended rule will be attached to the chain that are maintained by these green blocks whereas, the chain maintained by this older blocks will be only having blocks that satisfy properties of this set.

Now consider the opposite problem. Suppose you decide that I will ,this was my set of valid blocks. And now I decide to restrict it and say that here are some of the restriction. So I have a subset of blocks that I will consider valid. In this case, what will happen is that the blocks that are acceptable to the un-upgraded ones also have this property because this is a subset.

So any block that was here or here they are part of this larger set. So therefore, any node that was aware of this larger set will continue accepting blocks that are either here or here. However, when the nodes that are not upgraded, will start mining blocks according to the rules that only that does not put the block in this set, but rather only in this set, they will be not be accepted by others.

And very soon this nodes un-upgrade nodes will realize that the when they mine a block and they are not restricting their blocks within this set, they are being rejected. So very soon they will catch up, and then they will also upgrade. So that is a soft fork, because in this case, eventually there will be a single consensus change. So it was a fork for a while and then there was a single chain.

(Refer Slide Time: 08:35)



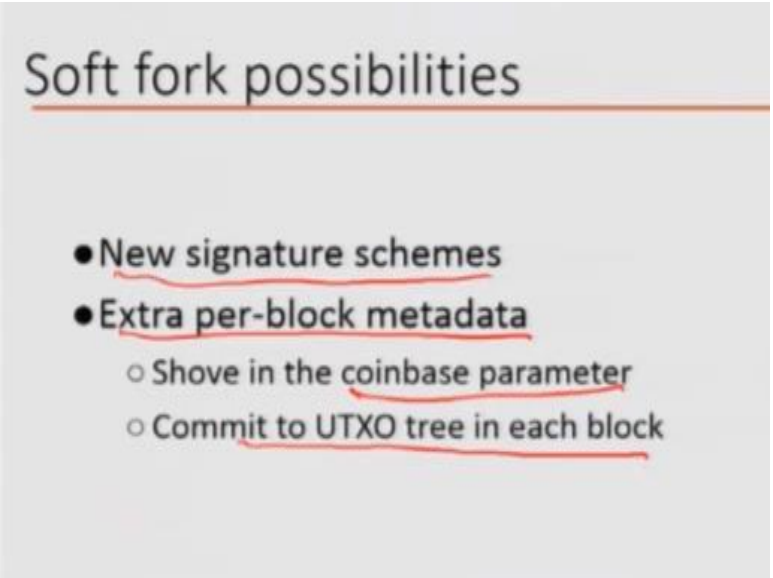
So soft fork example is earlier we discussed pay to script hash, where we said that the burden of writing the script was not with the sender of the money or the initiator of the transaction, but the receiver does a complex type of receipt of the bitcoin. So they write a script and provide the hash, right. So hash is like an address of that script.

So now when that was introduced, right so any block that contains a transaction of this type can also be permitted by the old the nodes that do not know about this, because they will only check the hash part, right. So see, it was an address here receiver address, that would be also a hash of 160 bits. And if it is an address of the script, that also is a 160 bit hash. Therefore, they will accept it.

And therefore, they will approve the hash, but they will not run the embedded script. So there, when they actually verify transaction, they might actually accept a particular transaction, which is not valid. But otherwise, if valid transactions are mined by the new nodes that are upgraded, they will be also permitted by the older nodes. So that is why it will be a soft fork and not a hard fork.

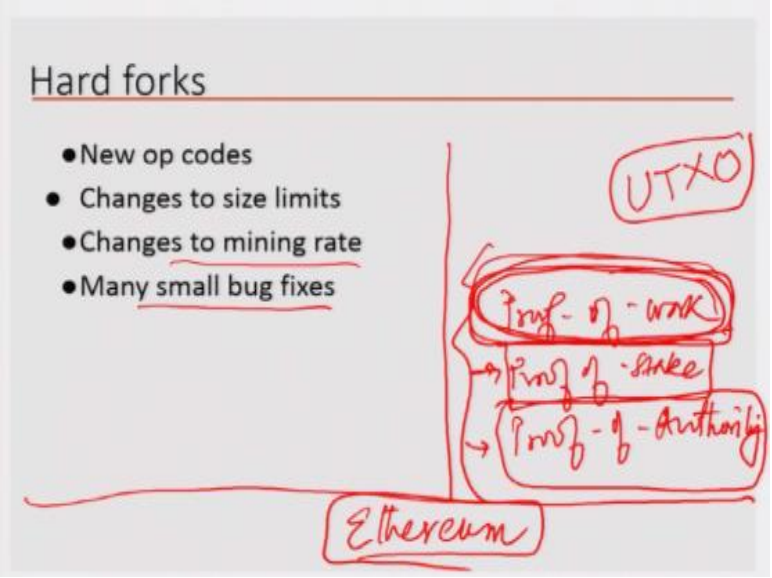
So this is how the pay to script hash was introduced in blockchain after the initial you know bitcoin was introduced.

(Refer Slide Time: 10:15)



So certain other things that will not require a hard fork or new signature scheme. So if you introduce a new signature scheme that could be fine. And if you start to require that in a block, there should be some extra metadata. That could be also be fine. You can actually put it into the coinbase parameter or you can commit it to the UTXO tree, the Merkle tree that you maintain, so it should not be a hard fork.

(Refer Slide Time: 10:45)



So what are the cases when hard fork is required? When new opcodes are introduced. So you have a new opcode so that this script cannot be run by nodes that have not upgraded because they have not they would not recognize the opcode. If you change the size limits of the blocks, right so then you need a hard fork because the large let us say you have changed it to a larger size, then you will the nodes that are not upgraded will start rejecting the blocks.

And so if you change the mining rate, or if you make bug fixes, remember that we talked about a bug in the multisig transactions, where it actually pops an extra data. So the data has to be replicated intentionally by the script writer. So if you try to fix that bug, there will be a hard fork and that is why it has not been fixed yet.

So what we understand from all this is that changing something in the blockchain in the block chain's code or block chain's, you know engine that, you know that executes the bitcoin scripts or the hardcoded limits, you have to go through a forking process. And that is quite expensive. Now this is true for you know most proof of work based blockchain or even in case of a proof of stake or proof of authority blockchain.

So let me quickly tell you what these other two means. So proof of stake means that it could mean a few different things depending on what is meant by stake. So in some cases, if you have, who is chosen as the next block maker, right. So in case of proof of work, you make it very democratic.

Well, I would not say democratic but you make it such that your chance of being the next block maker is proportional to the or your probability of being the next block maker is proportional to your computation power right your hash power. Now if you do not go by that, we saw that this is a very expensive process in terms of computation, in terms of energy. So let us say you do not want to do this.

So you may choose to have a different way of choosing the next block maker. So one possibility is that you make it proportional to the amount of coin holding by the different users. Now in bitcoin that is not possible because in bitcoin, we do not have a notion of accounts. So every time you make a transaction, you can use a different address and so all the information in the state of the blockchain is maintained in terms of unspent transaction or UTXO's right.

So you cannot do a proof of stake based selection of nodes based on how much they hold, because it is not clear how much the node holds because a node might be operating many public addresses. And you do not know that they belong to that

particular node. But in case of Ethereum, as we will see, in this session as we start looking into Ethereum, there is a notion of accounts and therefore each user has an account.

And all the coins that they actually own is actually consolidated in the accounts. Well, one can actually create multiple accounts with different addresses that is true. But then the stake will be based on the accounts. So now the question is that if you decide stake on how much money they have, then one that has more money will always be selected and that will not be fair, because then it will become a centralized system.

So people have devised other ways like for example, they have considered a mixture of how much money and what is the age of the coin like for how long they have been holding the coin, because one that is holding the coin for a longer time probably has more interest in keeping the money for a long-term investment. So that kind of thing is used as a stake.

Proof of authority is slightly different and this cannot be also done in a blockchain that does not have a proper notion of accounts or proper notion of some kind of a identity because proof of authority basically makes the chooses the next block maker from a set of privileged or authorized nodes. So not every node there, can be a miner like in bitcoin or in a proof of blockchain but nodes that are already chosen by some authority or some mechanism may be a voting process.

But some authority, so among those set of nodes which have the authority, there could be an algorithm like a round robin algorithm or some random choice algorithm or a mixture of stake and authority could also be used. So these are the different types of blockchains. Now certainly as we see here is that the proof of work is the most expensive one and also to make changes in the blockchain code or in the blockchain hardcoded numbers, proof of work will actually require a fork.

In case of proof of stake, it may or may not need a fork, but certainly in proof of authority, you can actually consolidate the changes in only in the authorized nodes and therefore, it may not require a fork. So but we will discuss that later if there is

time, if we get the chance to actually compare these different ways of deciding we will mine we will do the next block or we will create the next block.

And we will see that in nonpublic blockchains like hyper ledger and a few others Corda for example, none of this things are done. And instead there is a Byzantine fault tolerant consensus mechanism that is quite different from what these things are. Now that is where we will draw the line about bitcoin blockchain. We may come back to it once in a while to compare with the other blockchains.

But we have gone through the basics of bitcoin blockchain, the concept of transactions, concept of unspent transactions, the concept of consensus through proof of work or mining, the concept of blocks and concepts the cryptographic concepts required to show the authority or authenticity of the transaction sender. We also saw the notion of hashing and hash pointers to ensure the tamper resistance properties.

And we also saw the execution of scripts to actually make transactions which are more complex than ordinary transaction of sending just a few coins from one sender to a receiver. And we said that those are kind of smart contract but not exactly smart contracts. Now we what we will do is that we will go to our next example of a blockchain and which is the Ethereum, which was originally conceived with the hindsight of what is missing in the bitcoin blockchain.

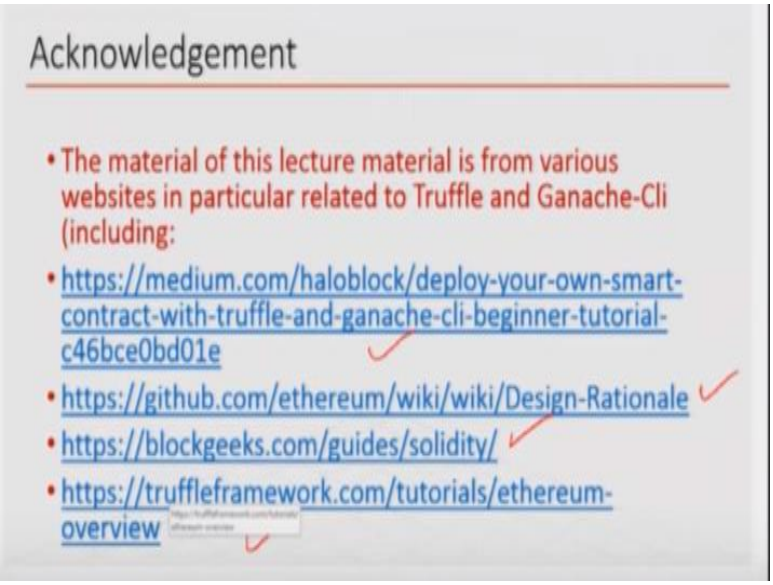
And then it got its own different ideas. One of the ideas inventors of Ethereum worried about is that a bitcoin blockchain is purely for cryptocurrency and purely for transactions. So if you want to store data permanently for example, let us say you want to store grades or you want to store medical information of patients or you want to store land record information, then we saw there are some ways like you can put them in the coinbase transaction as a coinbase parameter.

But or you can use a, you know proof of burn type of transaction. But it is not the most convenient and actually, bitcoin wants to discourage that kind of use of their blockchain. And that Ethereum wanted to address. So Ethereum wanted to create a blockchain infrastructure in which you can do all kinds of computation, not just

necessarily coin transactions or coin mining, although they also do coin transaction and coin mining.

But you can do all kinds of smart things with Ethereum. So let us now go and look at the situation or the concepts that are there in this Ethereum blockchain. So we will start in this session, a little bit about Ethereum. And then we will continue in the next session. So first of all, let me tell you, where you can find more material. So obviously in lectures you cannot learn everything.

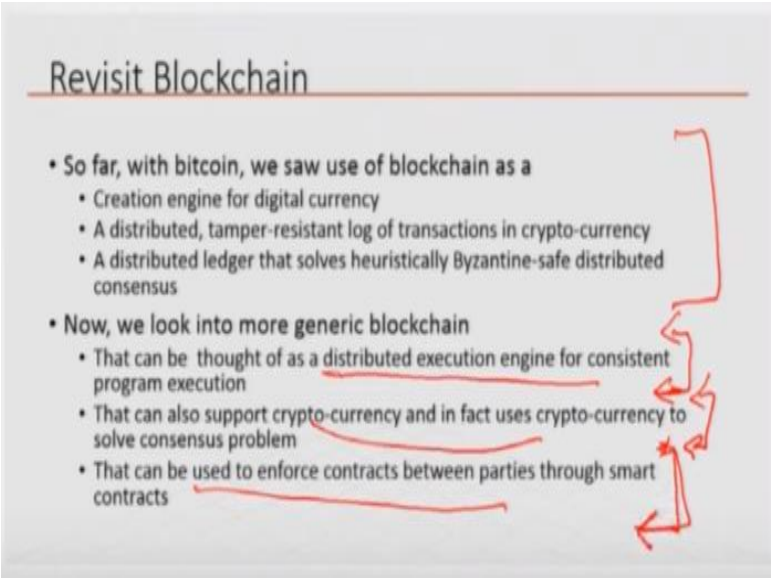
(Refer Slide Time: 20:53)



And so let me first say that we collected a number of concepts from various documents and blogs that we are going to present first about Ethereum and then we will go a little bit deeper and compare it with bitcoin, then move to private blockchain because in bitcoin there is no private blockchain concept but in Ethereum you can create your own private Ethereum blockchain.

And from there we will jump to more commonly used private blockchains. So these are the material that you should also read, because when we talk about them, a lot more detailed explanation of what we are talking about could be found in this documents.

(Refer Slide Time: 21:42)



So to understand Ethereum we need some context right. So far, we saw bitcoin only as example of a blockchain. And if you just know bitcoin, then you get the impression that blockchain and cryptocurrency are kind of synonymous. Because the bitcoin blockchain was a creation engine for a digital currency, and it creates a tamper resistant log, which is distributed and replicated at various nodes.

But this information is majorly about cryptocurrency transactions. And then the consensus mechanism there is under the assumption that any of the nodes could be malicious or any of the nodes could fail. And then how do we make sure that you can keep the ledger consistent across all replicas and that is what the consensus problem is all about. And we saw that proof of work and expensive method is used as a heuristic for Byzantine safe or Byzantine safe distributed consensus.

So that is what we saw in bitcoin blockchain. Now blockchain in the beginning of the course and repeatedly I have been saying is not necessarily about cryptocurrency and in fact, the cryptocurrency is probably be most harmful application of blockchain. But it has a lot more generic application and it is actually a very, I would say transformational platform in which computation, storage of information safely and integrity preserving way is possible.

So what Ethereum showed is that you can think of a blockchain as a distributed program execution engine for consistent program execution. So you can actually think of this as a platform in which there are many nodes and all the nodes are capable of

executing the same program and to verify that the outputs of the program are the same. And therefore, once they have verified that they can store that as a result of those execution.

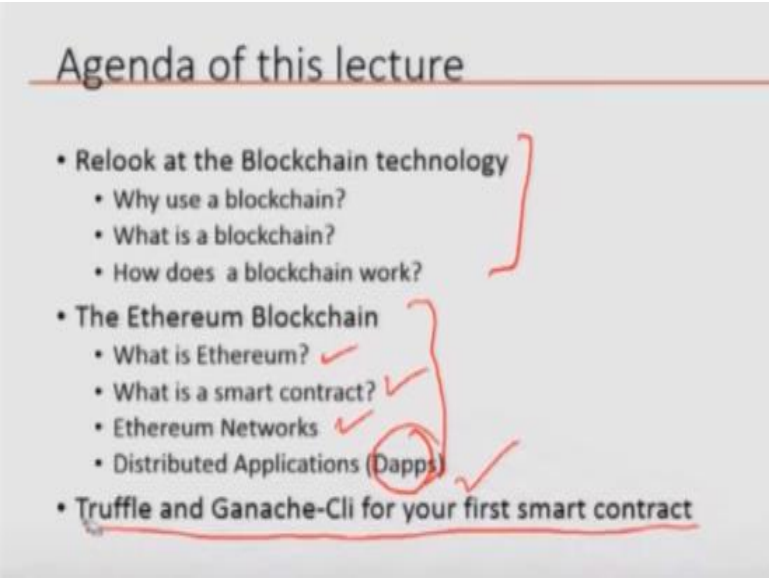
They can also store the code of the program so that in the future also they can verify that the execution results are in fact correct. But also it does support cryptocurrency, generates cryptocurrency by doing consensus, solving the consensus problem in fact using proof of work like bitcoin. And it can be used to enforce contracts between parties through smart contracts. So what is a contract?

A contract is usually an agreement between two parties, that if you do this, then I will give you this or if we do this together, we will get something from a third place together. And these contracts so far are usually enforced by a trusted third party. Otherwise nobody will agree to a contract. Either you trust each other , when, if you trust both sides, then you agree to a contract and you know that both sides will hold to the words of the contract.

If not, if you do not trust each other, then you have a trusted third party, who will ensure that the both sides hold the deal after the work is done, or after the conditions have been satisfied to fulfill the obligations under the contract. Here, we want to create a situation where there is no trusted third party and the platform itself enforces that the obligations of the contracts are met by all the parties involved, while the parties do not necessarily have to trust each other.

And that is what the smart contracts are all about. So a program execution, distributed program execution engine for consistent program execution, a cryptocurrency framework, like bitcoin, and then through smart contracts create a contract enforcement mechanism where parties do not necessarily trust each other.

(Refer Slide Time: 26:21)



So what are we going to do in the lecture about Ethereum? Compare or we will recall some of the things about blockchain technology. And then we will look into the Ethereum blockchain, particularly what exactly is Ethereum? What are the smart contracts? What are the Ethereum networks? And what are the Dapps or distributed application. This is a term that you will hear a lot in the context of Ethereum because distributed application is what we get on top of the smart contracts.

And then at the end, we will talk to you a little bit about some of the software that you will use like Truffle, and Ganache-Cli for writing your own first smart contract. Now remember that you can actually have a whole course on writing smart contracts, making sure smart contracts are correct, using various tools that actually check the correctness or kind of like, simulate smart contracts or even do some program analysis to check the security of smart contracts.

But that is not what we do in this course, right. So in this course, you will just get to understand what the smart contracts are? What do they look like? How do they execute? And get some basic principles in your mind and then it is up to you whether you want to run with that information and an introduction to some of the tools to self-teach yourself more about smart, especially Ethereum smart contracts which are written in a language called Solidity.

And then you can go with that and make yourself an expert in Ethereum. So when we come back, we will go on and look into blockchain technology in the context of Ethereum and then go into Ethereum.

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

Lecture - 13

Welcome back. So we were discussing about Ethereum blockchain. And to start the discussion on Ethereum we need to talk a little bit about why we use blockchain. And you will see throughout the course that whenever we say why we want to use blockchain, depending on which context we are talking about, which platform we are talking about, we will talk about it slightly differently.

So let us look at this in the Ethereum context. So earlier we said that, you know in the bitcoin we said that it is about generating cryptocurrency and then making transactions of cryptocurrencies, you know double spending proof and all that.

(Refer Slide Time: 01:02)

Now we are saying that blockchains are to be used when multiple parties perhaps located across the world need to share data and transfer value without trusting each other. Now we are bringing in data sharing as one of the key reasons why we want to use blockchain and transfer of value which may mean transfer of properties, assets or which could mean transfer of coins or cryptocurrencies, right.

So this is more generic than what bitcoin was doing. Now let us look at this trusting. So we are saying that, even if we do not trust each other, the different parties, we can

still make sure that we can securely and we can generate a trust that everything will be working fine without a third party ensuring that everybody is playing fair, right.

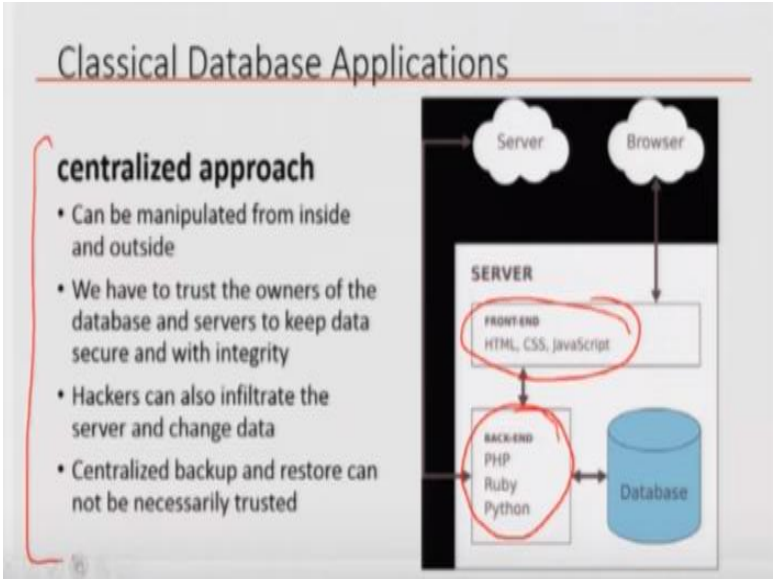
So usually we need a third party. Now this trust about the different parties in a transaction and when I say transaction here I do not mean cryptocurrency transaction, any world real world transaction like buying a property or sharing information and all that. So there is a, what the financial world calls this kind of mistrust is counterparty risk. Counterparty risk means that the thought or the risk that the other party would not hold up their end of the bargain, right.

So for example, I want to purchase a book from you, and I send you money but you do not send me the book, right. How do I ensure that that does not happen? Now currently, the way it works is that the other party which is selling the book is normally a large company, like Amazon and therefore they have a reputation to keep, because tied to their reputation is their value of the company.

Now if it is a small time person like Amazon Marketplace, then it is Amazon who underwrites the risk, right? So but in all these cases, the counterparty risk is mitigated by either reputational issues or by a third party. So the blockchains, such as Ethereum, tries to remove the counterparty risk through a clever use of mathematics, cryptography and peer to peer networking, right.

So that is what the blockchain platform provides is the remediation of the counterparty risk without a third party or centralized authority.

(Refer Slide Time: 03:52)



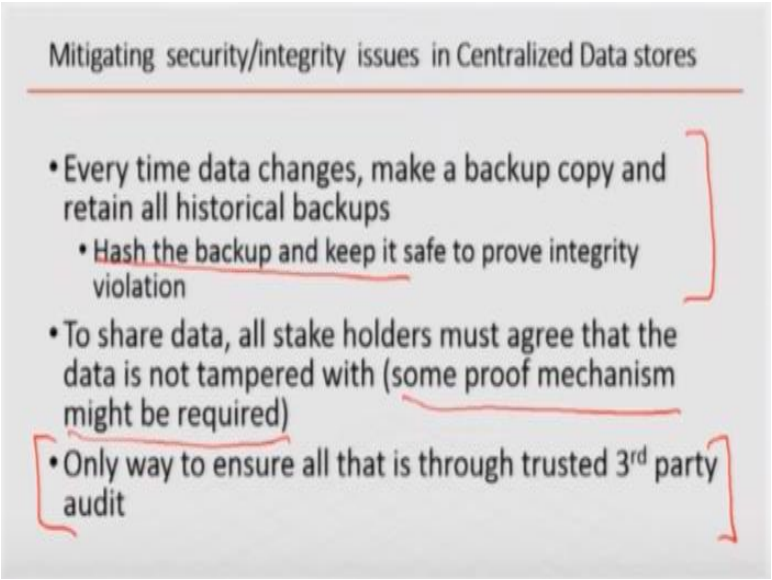
So let us see what is wrong with the centralized approach. And we discussed this before. Let us say the central database is your student grade database in a university and all the grades submitted by the instructors are in the database. And the way to interact with the database is through front end, like through a browser. And then the front end is actually rendered version of HTML, CSS, JavaScript, etc.

And then in the back end, there are some server program, server application like written in PHP, Ruby, Python, and so on. And then there is a database. Now the point is that how does the student know that the data in the database about his grades or her grades is the correct grade that the instructor provided? How does the instructor know that the grade they submitted is in the long term remains the same grade after some years.

Could be that somebody insider could have changed the database, or it could be that there was a cyber attack on the web server. And through that they gained access to the website, maybe through SQL injection.

And also when you backup and restore the database, depending on how you keep your backup, and who has access to the backup and so on the database backup, when you reconstruct the database from the backup, you may not be able to trust it. And this is the problem with the centralized approach.

(Refer Slide Time: 05:32)



So one possibility is that every time the data changes, you make a backup copy. And you retain all historical backups of the data. And you keep the hash of the backup and keep the hash safe to prove the proof integrity violation. But the question is, where do you keep the hash of the data because if the person who is in charge of the database hashes the backup and give the hash with him.

Then later on, if he decides to go and change the data, he will change the hash also. So the hash should be somewhere that he or she cannot manipulate, right. So one possibility is that you do not put all the data in your database in a blockchain, but you commit the hashes of the different backups to the blockchain. So every time you want to go and check integrity, you have to look the hash up in the blockchain.

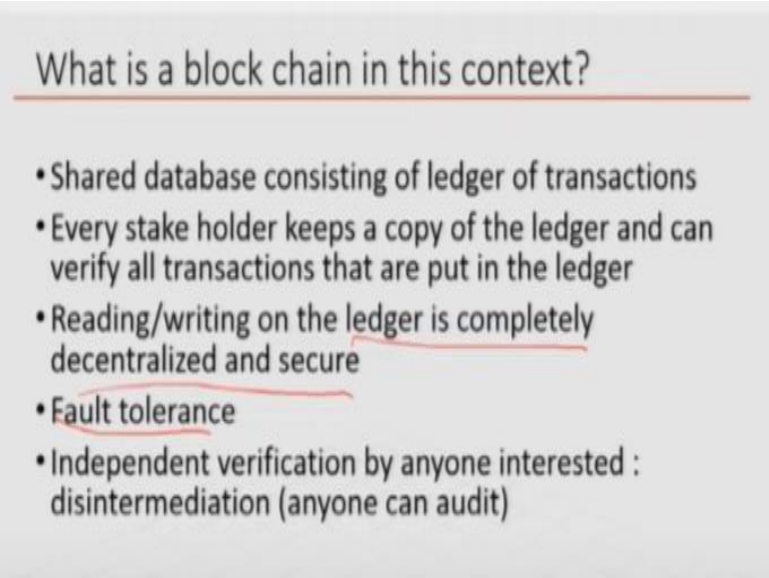
And given that blockchain is tamper resistant, that the person who is in charge of the database may not still be able to access the hash and change it. If he changes the data, and he cannot change the hash. So that is one possible way of mitigating this risk of a central authority keeping the data. Now the when you share data, all stakeholders has to agree that the data is not tampered with and maybe some proof mechanism might be required.

Maybe again some hash kind of thing is required to prove that the data integrity is maintained. So usually what we normally do in today's IT world is that we get certified by a third party auditors. And this IT system auditors come and check and say that okay I have checked all the logs and everything and nothing has been

tampered with and so on. I certify then whoever is interested, can actually look up the audit reports and satisfy themselves.

But this has all kinds of problems because the auditors might be bribed, the auditors might make mistakes, all kinds of problems. So I as the consumer of the data or whose data is being kept at the central server has no way to directly get a proof of integrity. You are always trust, you have to trust a third party auditor, or you have to trust the party that is keeping the data. And that is not very comforting, especially in today's day and age.

(Refer Slide Time: 08:20)



So one can consider blockchain in different ways. So one possibility is that you keep all the data in the blockchain. And every time you commit a data, that is a transaction. Every time you make a change in the data you make, that is a transaction. If every time you delete a data, that is a transaction. So all these things will be in the blockchain.

And since everybody has a copy of the ledger, so it is and therefore, depending on what consensus mechanism you use, you are unlikely to be able to do anything without being caught by some of the participants. So this makes the ledger secure, decentralized, and replicated and so on. So it is a fault tolerant, and anybody who wants to verify that the data has not been tampered with all the information about when the data was put, like initiated, that is a transaction.

When the data was modified, that is a transaction. When the data was deleted it was a transaction. So you can actually go and browse the blockchain, it is public, and then you can say that, okay you know I have a proof that I have direct proof, not by a third party auditor telling me that I have checked all the logs and everything. Now one problem with this is that data often is private, information like grades.

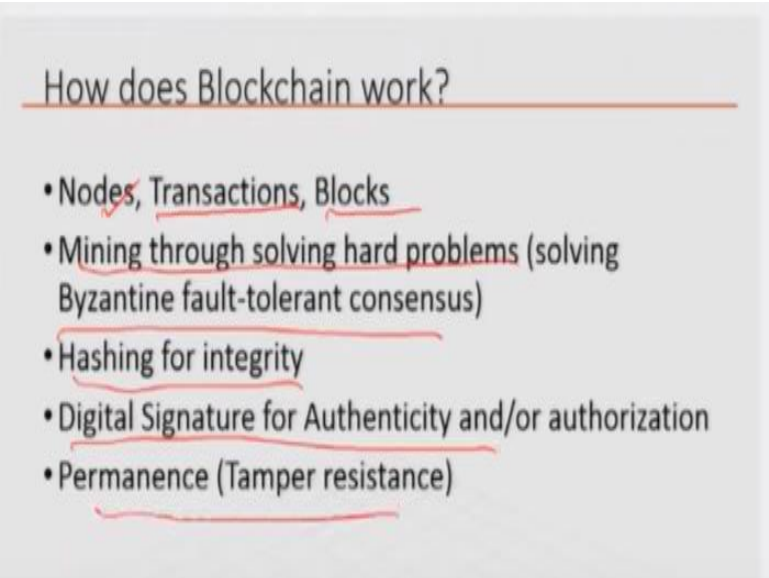
You cannot put grades of students in a public ledger. So in that case, you do not necessarily have to put the actual data in the blockchain, but you put the hash of the data. Every time you make a transaction on the data, you put a hash of the data in the blockchain. So every time you want to check data integrity, you have to check against the hash that is there in the blockchain.

So this may not make a whole lot of sense right now but we will talk about this later when we look at the applications. But the point is that blockchain is not necessarily for keeping all the data on the blockchain. Blockchain could be only about keeping the logs of all transactions in a secured form in the blockchain, so that you have proof that the data has been tampered with or not.

You do not necessarily but it in some cases you can, if the data is not necessarily public, or you cannot derive private information from the data by seeing the data, then you can actually directly put the data in the blockchain. But again putting the data in the blockchain will make the blockchain heavy and remember it gets replicated all over the places.

So you have to find out based on your application needs, whether you want to put all the data in the blockchain or you want to put some clever way of proofs of data tampering or not in the blockchain.

(Refer Slide Time: 11:22)

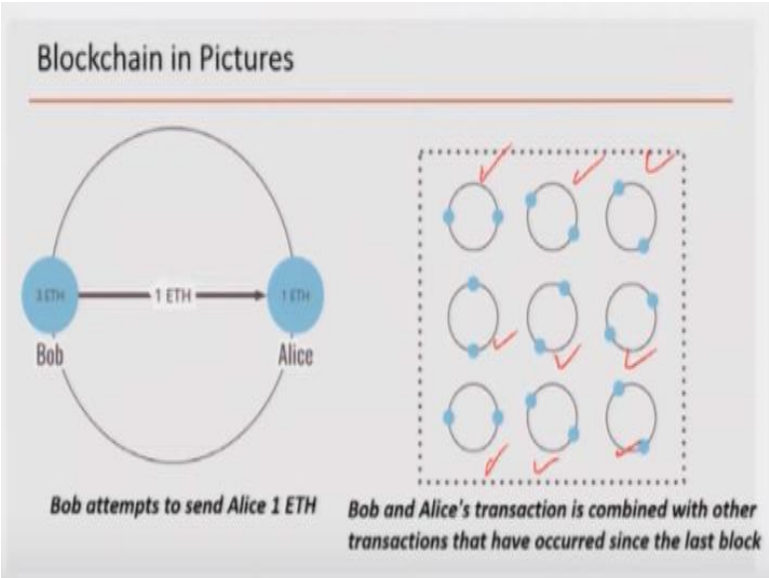


Now what are the concepts that are common in blockchain? We have seen the nodes which do transaction validation which do block validation which make blocks through mining or some other technique and when they make blocks, it could be that they might be solving puzzles or they could be part of the authority or they could be having enough stake to be one of the nodes making the next block or the nodes could be ordinary nodes who are just doing transactions.

So there is concept of transaction, there is concept of blocks and if it is through proof of word, then there is mining involved. And which is done through solving hard problems. Or if you can, you may actually do a Byzantine fault-tolerant consensus. We will see other blockchains where that is done instead of mining. Hash is used for integrity, digital signature for authenticity or authorization.

And there is tamper resistance. So we know all this. Just recalling some of the concepts.

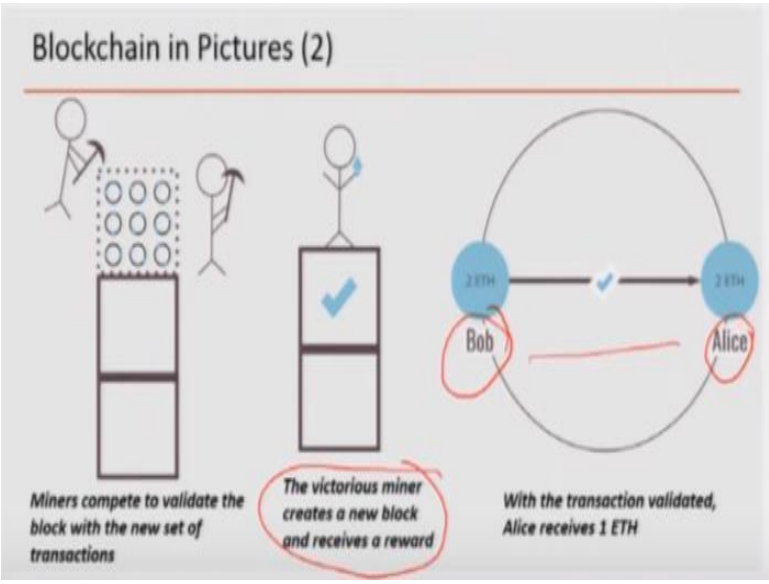
(Refer Slide Time: 12:29)



Now let us see one application, right. So let us say Bob has 3 coins. And he wants to send one to Alice and Alice has 1 coin. So at the end of this transaction, Alice should get 2 and Bob should get 2. That would be the next state right. So the current state is 3 and 1. The next state after the transaction should be 2 is to 2. But this is one of the transactions and here are examples of many transactions.

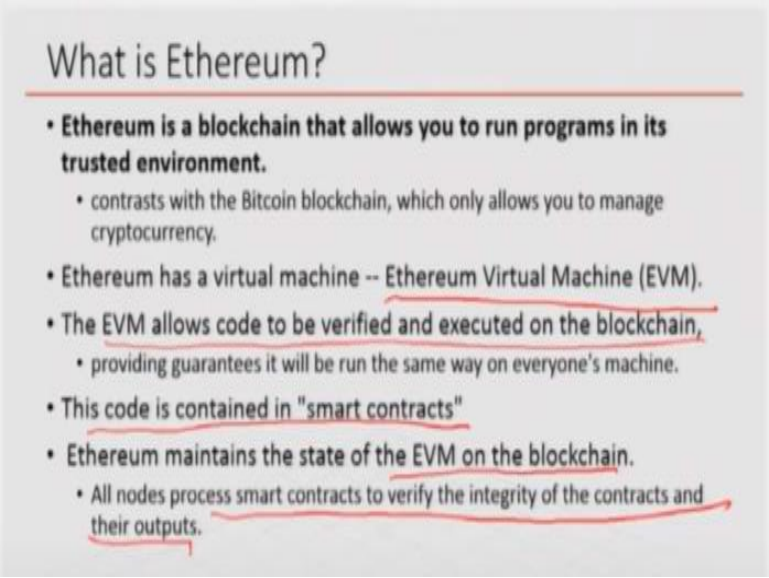
So when a node that wants to make a block gets enough number of transactions to fill a block, then they will do a mining or some other way of deciding that these are the transactions that I am going to put in the block. They will actually check validity of the transactions and then they will put the transactions and then attach it to the last block. So that is what happens here.

(Refer Slide Time: 13:24)



So at the end, when the block has been put into the blockchain and depending on what the process is, whichever miner wins, then the state of the situation changes. Now you have 2 and 2 for Bob and Alice. We know this from before, except that in case of bitcoin, we do not have a specific thing about Bob and Alice or this account and that account, right. So now we are talking about account to account transactions, right?

(Refer Slide Time: 13:59)



So that is something that is there in Ethereum and which is not in bitcoin. Okay, so now we are ready to see what Ethereum is really about. So first of all Ethereum allows you to run programs in its trusted environment, right. So we saw that in the bitcoin blockchain also we run scripts, right.

So every time I want to validate a transaction, I look at the input transaction and I look at the output transaction and I collect script from the input transaction, the output part of the output script of that previous transaction and the signature of this new transaction, I put them together and I execute a script. So that script execution actually ensures that the transaction is valid, right.

So here also we have such programs, which run every time a transaction has to be carried out. But this programs compared to the bitcoin blockchain are much more generic program. In fact, intentionally Ethereum made their programming language Turing-complete. So it can execute any arbitrary function unlike bitcoin, which was not Turing-complete.

And therefore, we could not run any arbitrary functions or write any arbitrary functions in that limited scripting language. We have programming language here, which allow you to run, create any programs for any function. And therefore, we can run any programs. And these programs run on something called an EVM or Ethereum Virtual Machine.

This Ethereum Virtual Machine is actually like any processor or Java Virtual Machine for example, which has a certain model of computation. So when it looks at the different instructions of the program, how this instruction is executed is based on the semantics of the Ethereum Virtual Machine.

So for example, in the bitcoin scripting language also we saw, remember we showed you how they have a stack and then and they put the data values on the stack and when they actually get to an opcode, depending on the opcode, they will retrieve the top one or top two or whatever number of arguments from the stack and then they will execute the opcode.

And then if there is an output of the opcode, then that is now put on the top of the stack. So that is what is the semantics of the script, right. And the semantics of the script is dependent on the virtual machine or the programming model of computation on which the program runs. So Ethereum Virtual Machine is also tag based computational model, where there is a stack.

The stacks actually are 32 byte words per entry. And you can have at most, I think 1024 entries in the stack so the stack can grow up to 1024 and as the opcodes are encountered, depending on the opcode you might take the top elements of the stack or the top two elements or top three elements whatever the opcode requires to execute, and then if the execution of the opcode has an output that output goes on the top of the stack.

Now Ethereum also since the stack is limited in size, therefore it cannot give you up completely Turing complete execution. So therefore, it allows access to memory and also it allows access to storage. So therefore, unlike the so if you have to be Turing-

complete and those of you who have taken a course in theory of computation, know that a Turing machine has an infinite tape right.

And because it has an infinite tape, you know you can have arbitrary long intermediate state of the program. So here also there is memory and there is stack and there is storage so you can actually do anything you want and your state can go pretty large. And then again it may go become small, but that is how the Ethereum programs are executed.

So Ethereum programs when they get executed, they produce some result. And depending on what the program is supposed to do, this program could be putting money from one account to another or it could be launching a new contract, a new Ethereum program, a smart contract or it could be actually doing some data storage. So depending on what the program is supposed to do.

Now what happens is that so a transaction is actually involves invocation of functions in a smart contract. So smart contract is more like if you know C++ or Java, it is actually a collection of methods and some data, right. So when a transaction needs to happen, it may refer to a smart contract and some method or function inside the smart contract.

And it can do that you know a transaction can be quite complex. So it can actually do multiple different calls to different functions and in different smart contracts and these have to be executed. So when the transaction has to be validated, every node that is validating the transaction will also execute the entire code right. So every node that is validating a transaction will actually run code on their EVM.

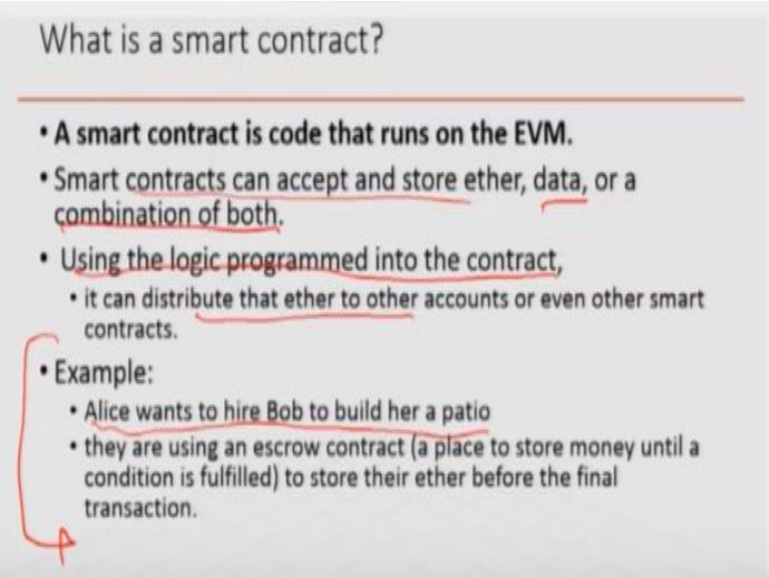
And this programs therefore, have to be what we call deterministic programs. So a deterministic program is a program that no matter how many times you run they will always produce the same result. Unlike a non-deterministic program or a randomized program, where different runs may produce different results.

So therefore, all the different nodes that is verifying the transaction or validating the transaction, when they execute, they end up with the same final state and final result.

So the code is contained in smart contracts. And the state of the EVM is persisted on the blockchain. So every time you execute a smart contract, or a transaction, you might change the state of the EVM. And that information is put on the blockchain.

So it is not just that the transactions are on the blockchain, but the EVM states are on the blockchain. So all nodes process smart contracts to verify the integrity of the contracts and their outputs.

(Refer Slide Time: 21:32)



So as I said, a smart contract is code that runs on EVM. Most of the smart contracts are written in a language called Solidity. Solidity is an object oriented language, kind of like JavaScript. So smart contracts themselves can accept and store ether, right? So it is not only that a person's account on Ethereum can store ether and spend ether. But smart contracts themselves can also accept and store ether.

It can also accept and store data, or it can accept and store both data and coins. Based on what is written in the smart contract, in the code of the smart contract, it can actually also distribute that ether to other accounts, other human accounts, or to other smart contracts. So kind of like each smart contract as if they also have an account, right? So humans can create accounts as players.

In the account they can put ether, they can earn ether by selling stuff, and then somebody sends ether to their account, or they can mine ether, if they are if they have enough computation power, and they can put those ethers in their account. Similarly,

the smart contracts themselves also can get ether, can spend ether and they can get data and store data, then remove the data and so on.

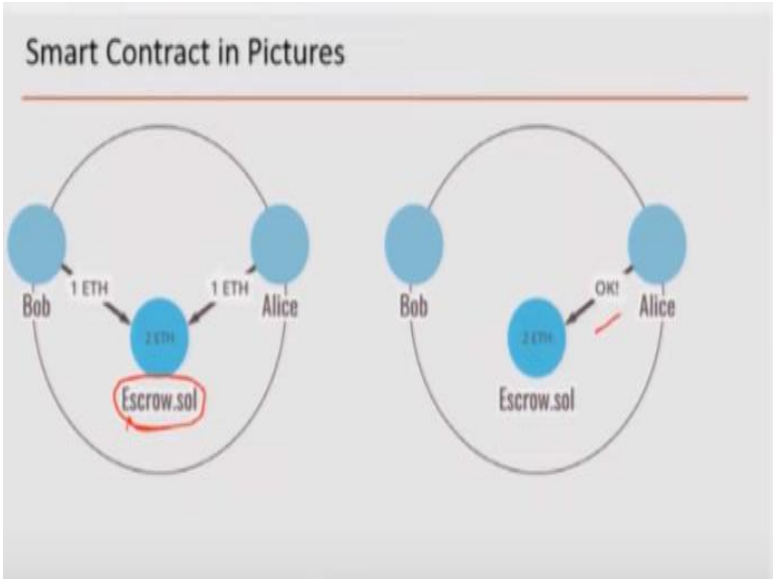
So let us see what kind of transactions are done with smart contract. And here you will not see much different from what this smart contract that we are going to show now is very similar to what can be what we have seen before in terms of escrow transactions, in case of bitcoin. Because this is this does not require a lot of programming primitives. It can actually be done with very simple programming primitives, like what is available even in bitcoin.

So let us say Alice wants to hire Bob to build her patio.But Alice does not want to give the money upfront to Bob because she is afraid that Bob will then do a poor job or not do the job or do worse, make her patio even worse. So Alice tells Bob that you deposit some money to an escrow account and I will deposit also money in that same escrow account. And an escrow account is basically a contract.

So contract, which is so instead of a person, like a Judy, like in case of the escrow we saw in case of bitcoin where there was a third party Judy, who was you know kind of working as an escrow, here a program will work as an escrow. So the program is written in such a way, so that if the contract be the real world contract gets fulfilled, for example, Bob really builds the patio to Alice's satisfaction, then Alice can invoke a function in that contract, so as to pay Bob the money.

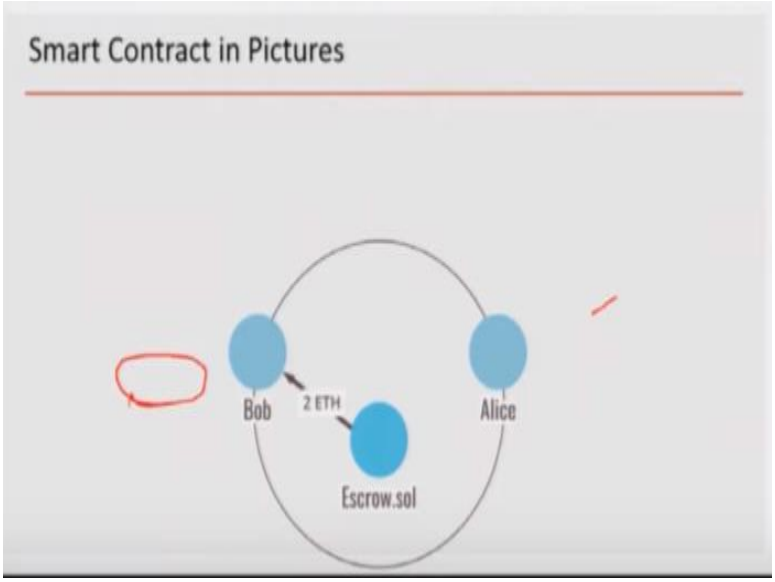
And if Bob does not do a satisfying job or does not build her patio, then she can invoke a function in the contract to actually not only get back her money, depending on what is written in the contract, maybe get Bob's deposit also. So that is how the contract. So here instead of a human third party, we are using a program to enforce this contract.

(Refer Slide Time: 25:14)



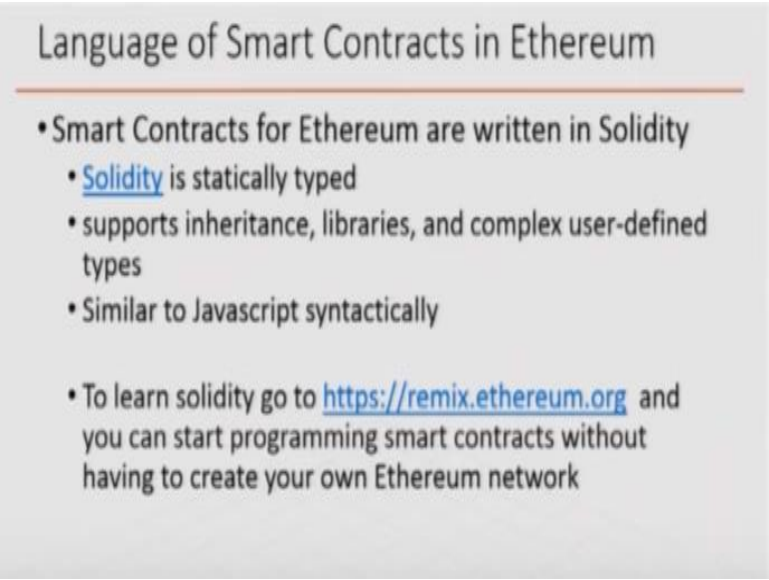
So here is an let us say an escrow smart contract. And you have both of them put some money in there, in this case 1 ether each. And then at the end, Alice says, sends an okay message to one of the functions that pays up Bob 2 ethers, right.

(Refer Slide Time: 25:33)



So one, his deposit back, and one for what he has earned.

(Refer Slide Time: 25:39)



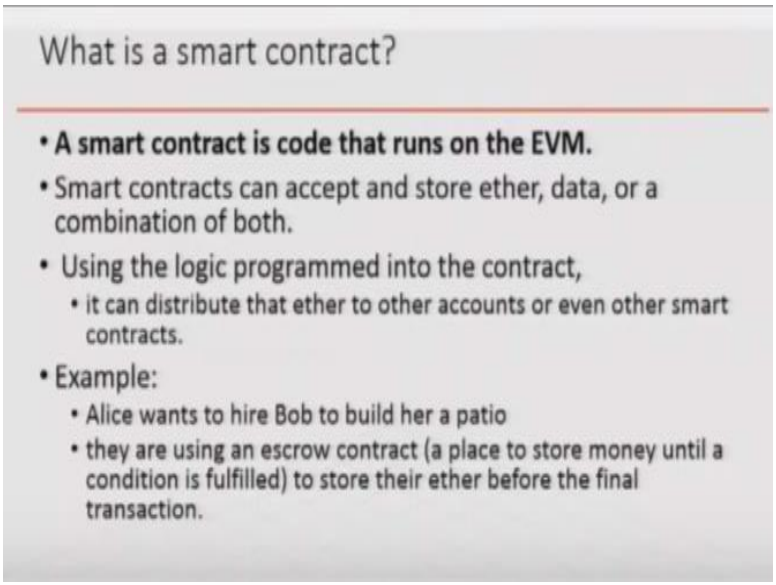
So as I said before, the smart contract of Ethereum are written in Solidity. And in the next session, when we come back, we will talk about a little bit on Solidity, what it is like and so on. And I will give you a pointer as to through which you can actually start writing your smart contracts in Solidity in a simulation environment. So that is where we will start in the next class, in the next session. Okay.

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

Lecture - 14

Welcome back to the course on blockchain applications and techniques. So last time, we stopped at this point when we were discussing smart contracts.

(Refer Slide Time: 00:29)

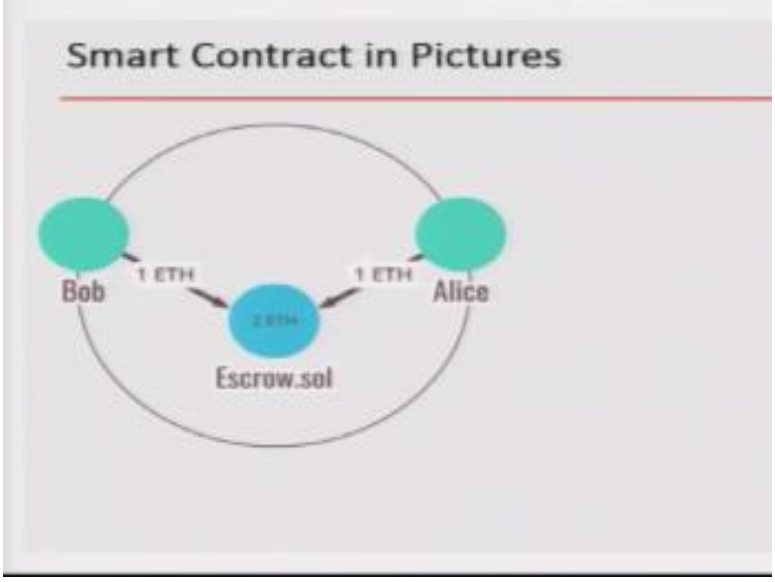


So a smart contract, as we discussed last time is a code that runs on the Ethereum Virtual Machine. And they can accept and store ether, which is the currency in the Ethereum and data, or a combination of both. And the contract is just like another program written in a very specific language. And also contracts can create or deploy other contracts. And also it can distribute ether or send data to other contracts.

So we were talking about this example last time. So here Alice wants to hire Bob to build her patio. And now Alice does not want to pay upfront in case Bob fails to do the work to her satisfaction. So they create an escrow account, which is basically a smart contract, and in which they both store equal amount of ether. So Bob puts in some money as some kind of a deposit and Alice put some money, which will be eventually paid if the work is done to her satisfaction.

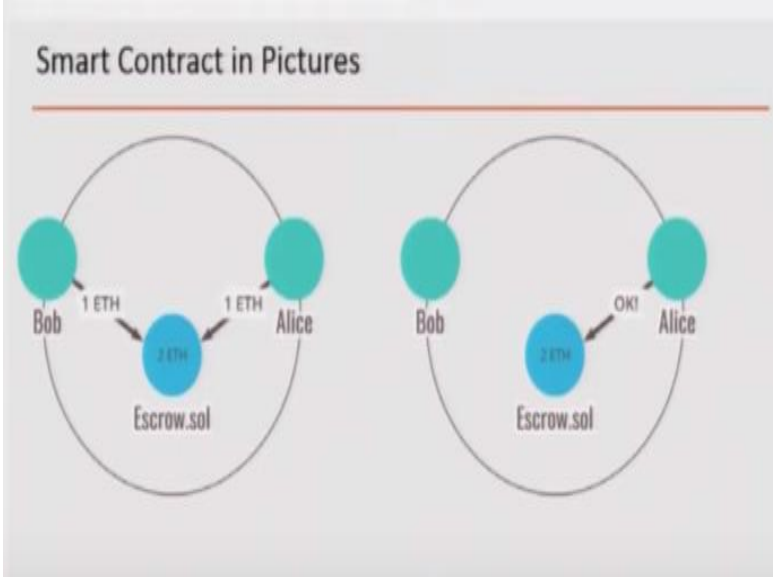
And the smart contract is programmed to accept a command from Alice that says that I am satisfied and now you can pay.

(Refer Slide Time: 01:56)



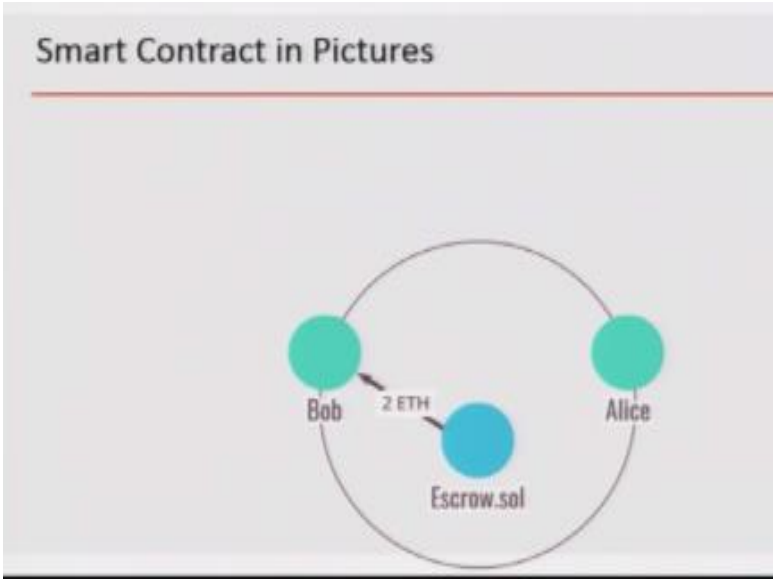
So pictorially this is what it looks like. So Escrow.sol is a solidity program which is a smart contract which will be deployed on Ethereum and Bob, basically and Escrow.sol has functioned to accept ether. So Bob will present 1 ether and Alice will send 1 ether. So now we will have 2 ethers in the escrow.

(Refer Slide Time: 02:22)



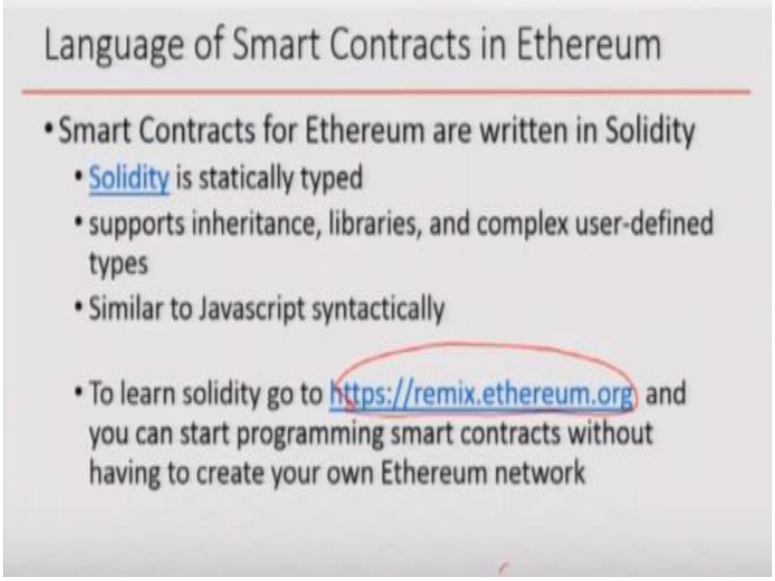
And then Alice once Bob builds the patio and Alice is satisfied, then Alice will send a message, which is basically a function call, and then the money will be paid out to Bob.

(Refer Slide Time: 02:34)



So Bob will get back his deposit as well as Alice's payment. So this is how the smart contract works.

(Refer Slide Time: 02:42)



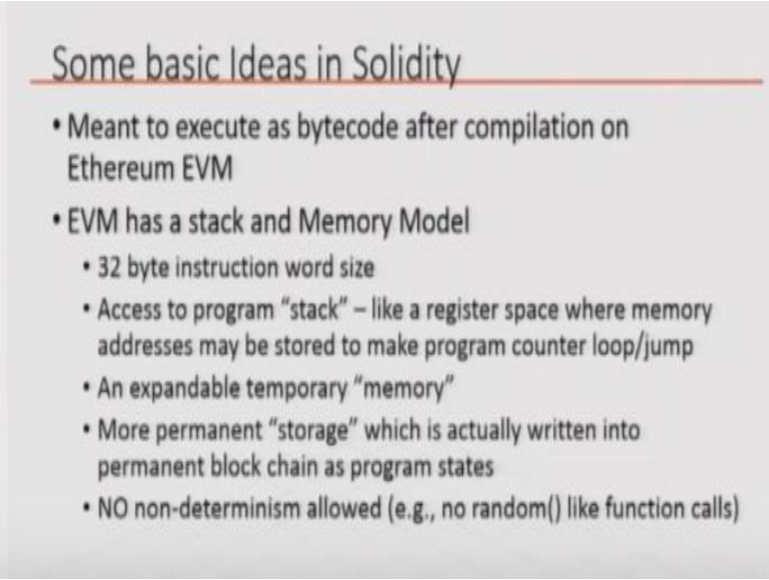
Now the language of smart contract is in cases of Ethereum is called Solidity. And so Solidity is kind of like an object oriented language, and it supports inheritance. So you can inherit from given contract, you can inherit and enhance the contract. It has libraries and complex user-defined types. And syntactically, it is closer to JavaScript.

And as I said, in the very beginning of this course, in this course, we are not teaching, really smart contract programming. But I will give you some pointers as to where you can start learning about Ethereum smart contract programming. So you can, for

example, go to this place, remix.ethereum.org and you can start programming smart contracts. They also have some examples, smart contracts already there.

And you have to, it has a test network, so you do not have to start your own Ethereum network. But very soon, we will also see how to use ganache-cli, which is a command line interface for creating a test blockchain, Ethereum blockchain and also create contracts using Truffle. And you can actually use that for doing some programming practice for Ethereum smart contracts.

(Refer Slide Time: 04:12)

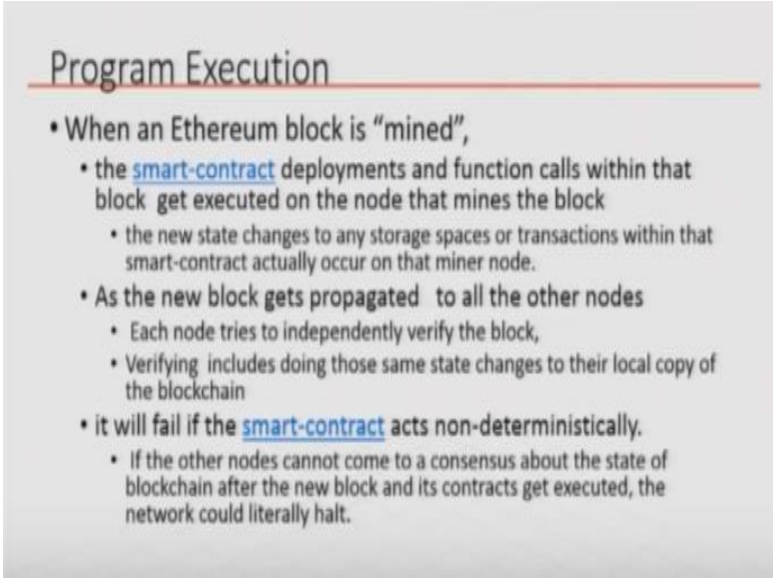


So some basic idea about Solidity. So Solidity when it is compiled, it becomes bytecode, which is going to run on Ethereum Virtual Machine or EVM. And the EVM has a stack based memory, stack based execution model. So it has 32 byte instruction word size, so 32 times 8. It is exactly 256 bits long. And there are registers which are so stack is that of registers. And these registers are basically 256 bit wide.

And you can store memory addresses, and make program counter loops and jumps and so on. And it has some expandable temporary memory and it has permanent storage. And what happens in the permanent storage is basically what goes into the blockchain and what happens in the memory are temporary and they do not make into the blockchain. And one important thing about Solidity is that it has no construct through which you can make non-deterministic programs.

So because if you have non-deterministic programs, then these smart contracts, the validation of the execution of smart contracts cannot be made by all the nodes because every time you execute it, it may non- deterministically give you different results. And therefore, non-determinism is not allowed in this language.

(Refer Slide Time: 05:46)



So what happens is that, when you make, you know smart contract, make function calls to smart contract, deploy a smart contract, these are all considered transactions. So after some transactions a miner would like to put them all together in a block. And then the miner will execute all the smart contract functions and deployments that you have met. And then record the state change to any storage space.

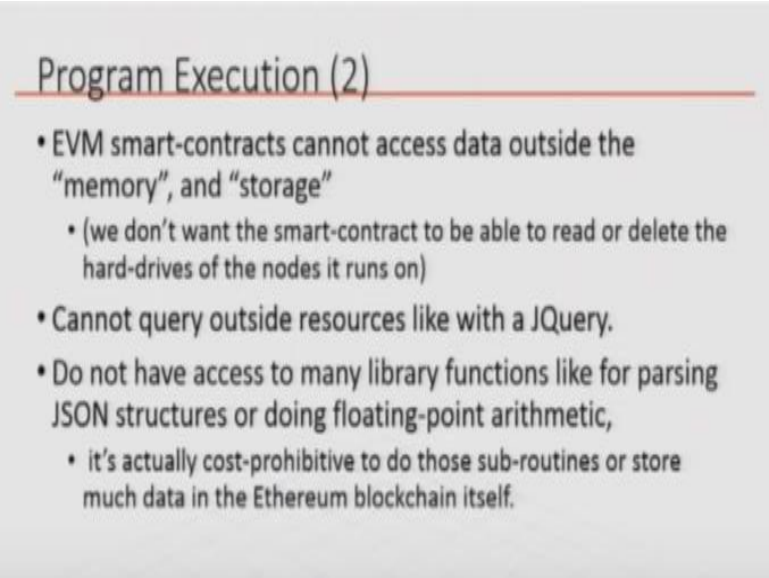
And all the transactions, these things are recorded in the block that it creates. Now once the block is created in a proof of work environment, it has to also solve the hash puzzle. So multiple nodes might be trying to do the mining at the same time, but whoever solves the hash puzzle will be actually the winner and it will have the same kind of issues as bitcoin that multiple miners may also make the miner block.

And solve the hash puzzle almost at the same time, and then the propagation will take place. The block will be propagated throughout the Ethereum blockchain network. And then eventually one will win. And then the other, there will be a concept of an uncle block in Ethereum which is absent in bitcoin. Now as the blocks are propagated to various nodes, each node tries to independently verify the block.

Now verifying means that you have to also do all the executions and make the same state changes. And that is where non-deterministic behavior would have if there was non-deterministic constructs in the Solidity language and then there would be problem because all the nodes when they execute the functions, if any of them are non-deterministic, it might give you different result.

And then they would not be able to validate the block and this will make Ethereum network fail to actually mine the block. So therefore, this is a good reason why non-deterministic construct is not allowed.

(Refer Slide Time: 07:59)



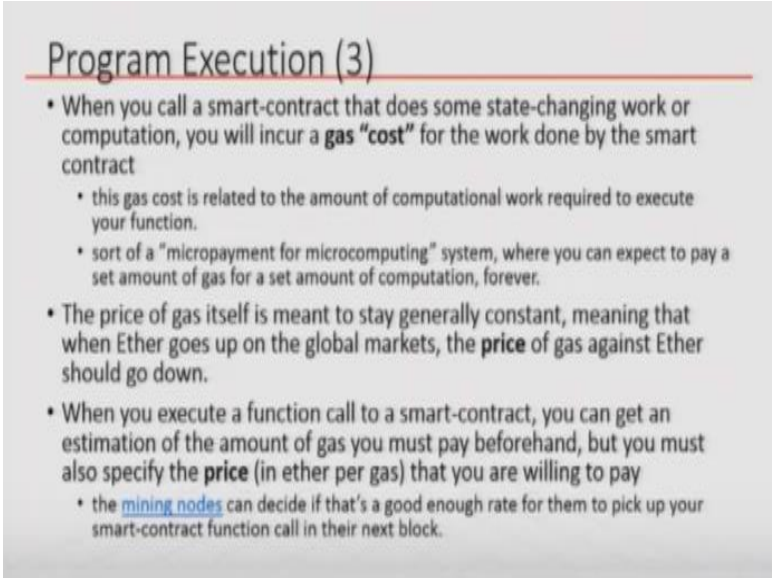
Now EVM smart contracts cannot access data that is outside its memory or storage. So it would not be able to read information from hard drive or other resources on the machine. And it would not be able to also delete or do anything. Because then if that was possible, then there will be no the smart contracts are deployed anonymously by users.

So if this could actually go outside the EVM and interact with the machine on which you are doing this mining, then you could get a lot of cyber-attacks and therefore, it is not allowed. So it cannot query any outside resources, like using a JQuery kind of libraries. And you do not have access to library functions like parsing JSON structures or doing floating point arithmetic, because it would be very expensive to do so.

And then we do not want the execution to be expensive and as we know we already discussed this that Ethereum actually charges per instruction execution, it charges you gas and gas is priced by ether. And therefore, when you actually create the contract and make a transaction, you have to assess how much gas is required for executing the transaction. And therefore, there is a limit on the gas.

And the one thing about the gas is that because of this limited amount of gas that you upfront, put up that you are willing to spend, therefore, infinite loops are not possible, because eventually you will run out of gas.

(Refer Slide Time: 09:42)



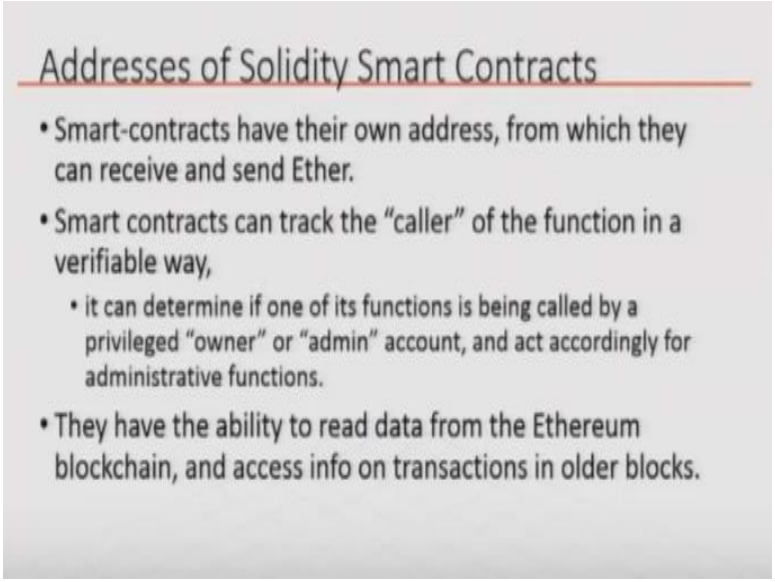
So when you call a smart contract, and it does state changing work or computation, you incur gas cost as I said, and the gas cost is related to the amount of computational work that is required to execute the function. And so you can think of this as kind of like as micro payment per micro computing operation. And you have to upfront do this. Otherwise, as I said that there may be an infinite loop.

So you have to set an amount of gas for a set amount of computation. You might also have to mention how much price you are willing to pay for the gas and this is the way you might actually prioritize your transaction over other transaction because when the mining node has to decide which transactions to put in the block, it might prefer the ones which are willing to pay the more price per unit of gas.

Because you are already telling that you are willing to put in so much amount of gas against the ether balance you have in your account. And so the gas amount you are fixing up front and therefore the price is the one that you can vary that will actually give you the ability to prioritize your work. So there are ways to estimate the amount of gas you must pay beforehand.

And you must also specify the price and the mining nodes decide based on the price you quote whether to take your function call into their next block or not.

(Refer Slide Time: 11:24)



So addresses of smart contracts is an interesting part of this Ethereum blockchain. As you know that in case of bitcoin, we did not have any smart contract. We had bitcoin scripts, which were used to do transactions, and we had some ability to do slightly complex transactions like you know escrow transactions, for example. But there was no named entity for and permanently residing entity for this scripts.

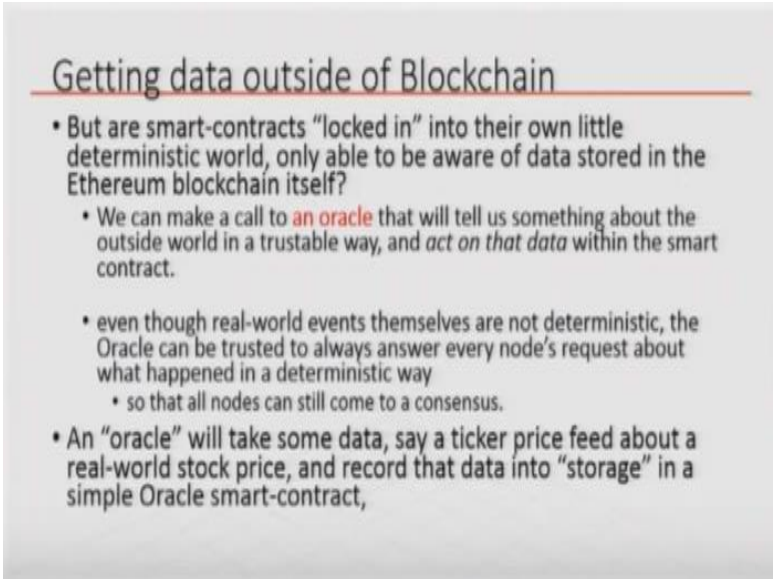
Whereas in case of Ethereum smart contracts are entities that live on the blockchain. Once you deploy a smart contract, it is on the blockchain. And therefore, this smart contracts also has the ability to accept ether. So smart contracts has balance. And in the balance, you can transfer ether, you can transfer out ethers to another externally owned account, which is basically the accounts that have a human wallet or something.

But then also there are smart contract accounts, which are actually self -managed accounts and so they have their own addresses, like any other Ethereum account address. And the smart contracts can also track who is calling the contract.

And therefore if the contract caller has a particular function caller has a particular type of privilege like the owner of the subcontract the person who deployed the subcontract or an admin account, then it can also do certain administrative functions which cannot be done by a regular user of that smart contract. And the smart contracts also can read data from the Ethereum blockchain itself.

And therefore it can also access transactions from old blocks. So it may want to check whether a particular transaction has taken place. And then based on that the its program logic might be that based on that, it will do something. So it has the ability to read that. But that is it like it cannot go outside the blockchain and go into the resources of the machine itself outside the blockchain or in the contract itself.

(Refer Slide Time: 13:50)



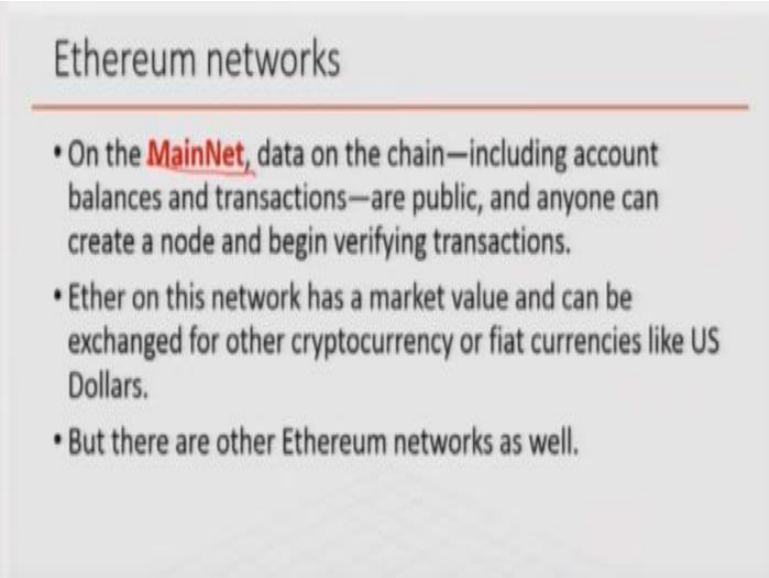
So are smart contracts locked into their own little deterministic world and not be able to get data outside from the blockchain? So this would have been a problem. For example, suppose your smart contract is based on the stock market transactions or it might be based on the weather. And these are the information that is not residing on the blockchain. But more importantly, this data is very dynamic, right?

So the price of a particular stock at this point may be something and it might be something else after a couple of seconds or microseconds depending on the transaction speed on that stock exchange. But Ethereum solves this problem by using something called an Oracle contract. So Oracle contract actually can get data from an outside real world event.

And this data if since it is a changing data, so you might think that this will be non-deterministic, which means that when an action is taken by a smart contract based on let us say a stock price, and then when the block percolates to the other nodes and other nodes are going to check that, it might actually get a different value and then it will be a problem. So therefore, the Oracle actually copies values and keep it stable.

So that determinism is preserved and therefore consensus does not get into a big problem. So this Oracle is a smart contract that has storage and in that storage, it stores data, maybe it will record it with the timestamp. So it will actually get that data at that particular timestamp when that block was created. And that is how this determinism is preserved.

(Refer Slide Time: 15:49)



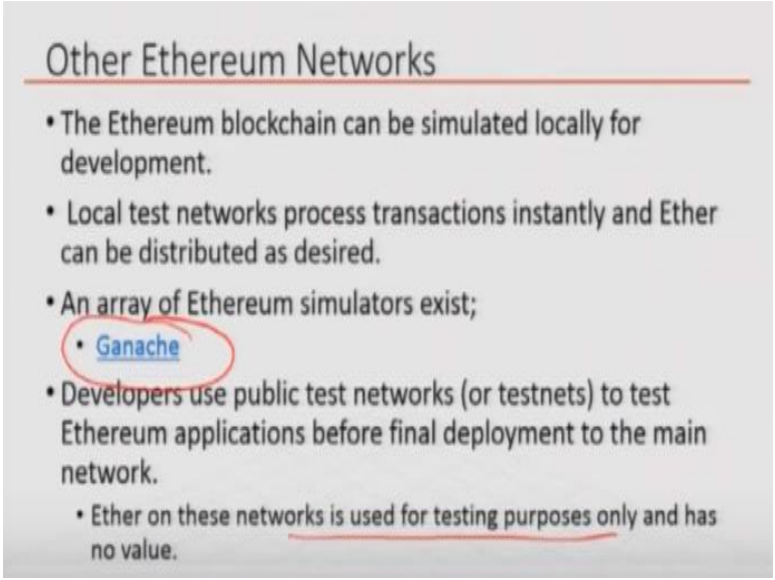
So Ethereum has multiple different network. So blockchain for example, has a single network, right. So of course, you can create a test network also for blockchain. But Ethereum Foundation actually maintain multiple different networks. So the MainNet, which is the net on which real Ethereum transactions happen, and this network actually has ether transaction all the time.

There are exchanges, people's wallets and everything are nodes on that network. And therefore, as various speculations and other things drive the price of the ether that applies to the MainNet. But you can also create other test networks in which the ether is not really prized by real world currency. And there people use that kind of test nets to test out their smart contracts, whether it works or not, because there the ether does not cost anything.

So the MainNet has all the data that is public, like blockchain data. And anyone can create a node and begin doing transaction. They can also do verification of transaction. They can do of course the mining with proof of work will require a lot of computation power. But to join in the MainNet and start buying, start getting ether, from exchanges and then spending ether on purchasing stuff, all that stuff you can do through this MainNet.

So this ether has market value, and you can go to an exchange and get money, you know make a transaction to give your ether to the exchange, and then you can get real current fiat currency, like US dollars against those. And there are other Ethereum networks that are test networks.

(Refer Slide Time: 17:50)



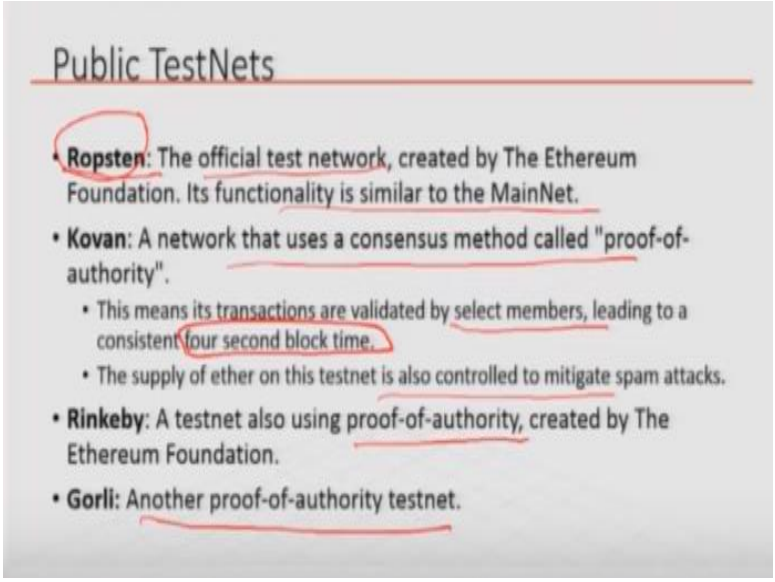
We will show you some of those. Now Ethereum blockchain can also be simulated on your own computer, right because when you are doing development, you are not going to first try it out on the Ethereum MainNet, because if you deploy a contract

which has bugs, then it will be there forever. And then it can be exploited in many ways. If you create a smart contract with some ether in it as a balance, then with a bug, those ethers may be stolen and other things can happen.

So therefore, you have to do a local test network on which you can do deploy smart contract, start doing transactions on it and test out, you know all its functionalities. So there are lots of Ethereum simulators that have been available. So the one we will talk a little bit about today is Ganache Ethereum simulator.

And, but more advanced developers, they actually can use public test networks or testnets to test Ethereum applications before final deployment. And ether on these networks are used for testing purposes has no market value. And that is why it is safer to test out your programs on this kind of network rather than on the MainNet and then once you are reasonably sure about the network, then you can start using the MainNet.

(Refer Slide Time: 19:23)



There are certain ones that you can find on if you go to Ethereum.org. You will look for testnets and you will find at least three, four such testnets which are run by Ethereum Foundation. So Ropsten is one such testnet. And it is actually an official test network and directly created by Ethereum Foundation. And it is very similar to MainNet. It is a proof of work type of blockchain.

And so everything that you are going to do on MainNet can be well tested on Ropsten and in Ropsten you can get ethers for free. Those ethers have no value, so you can do

whatever you want. Kovan is another network that uses a different consensus mechanism rather than proof of work. It is called proof of authority. So proof of authority has certain privileged set of users who are going to validate the transactions and who will be mining the transactions.

So this select members have some identity unlike the regular users who can create a private key and public key and use the public key hash for its wallet address, and then it can use the private key to do signatures and all that stuff. But the select members will have to have very significant amount of proof of their identity and therefore it is faster to do transaction validation because these are people you know.

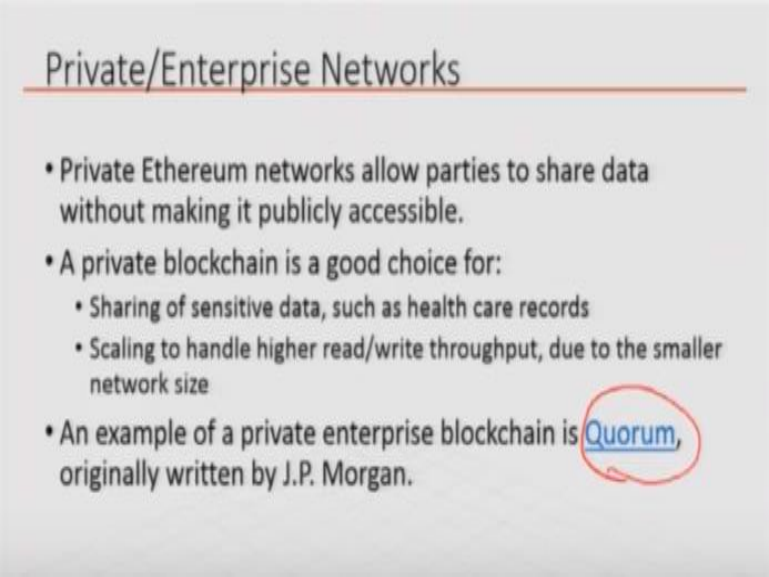
So if they do, if they put as invalid transaction in a block or if they try to approve double spending kind of transactions, you will eventually catch them. So that is a different kind of consensus mechanism. So the proof of work consensus mechanism does not trust anybody. So everybody is equal in the network. And there anybody can mine and validate transactions.

Whoever has more computation power is likely to be the eventual miner of the next block. But that method is very expensive, time consuming. So therefore, the proof of authority since only a select few members with strong identity proofs are going to make the validation. So therefore, we are going to see a mining and faster transaction time. So four second block time is what we see in the proof of authority type of cases.

The supply of ether on this net is also controlled to mitigate spam attacks. So this is another thing that is different from Ropsten other than the consensus mechanism. Rinkeby is also another testnet which also uses a proof of authority. And also it is created by Ethereum Foundation, same kind of concept. And Gorli is another proof of authority testnet. So there are multiple different type testnets.

If you are going to use the MainNet which is still a proof of work, then you want to use Ropsten for your test. But if you are going to use a different proof of authority type of blockchain eventually for your transactions and for your contracts, then you might use some of these. So these are still not part of the MainNet, these proof of authority. So therefore, these are more currently in the development stage.

(Refer Slide Time: 23:02)



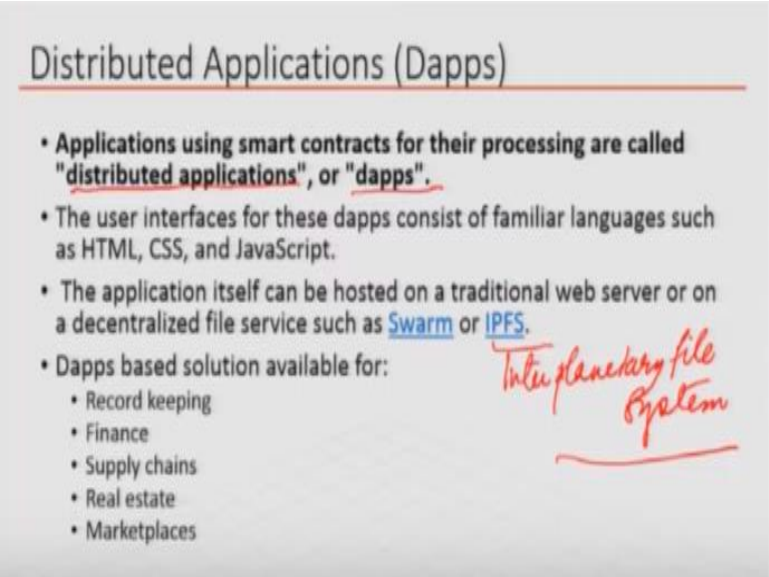
Now another thing people are doing is to creating a private Ethereum network. So let us say in your organization, you want to share data or between couple of organizations like multiple banks, you want to create a blockchain, but you do not want to make your transactions, your smart contracts public. So what you can do is, of course you can choose a different permissioned blockchain, which we will be discussing later.

But in a permissionless kind of set up, and with the support of a cryptocurrency mechanism on top of it Ethereum is a good choice for making a private Ethereum network. And then the idea there is that you create the Ethereum which is different or unconnected to the MainNet or any of the test networks. And then you actually can use proof of work, if you do not want to give everybody an identity. You can use proof of authority,if you can provide some of the accounts as privileged accounts who have the authority.You can do also proof of stake, you can choose. Now you can share sensitive data, such as healthcare records, because healthcare records, you do not want to put on a public blockchain because that violates privacy. And in case of countries like US, there are strong laws against making healthcare records public.

And also they scale well when get high throughput, because you are not you know using a huge network with millions of nodes connected and therefore it will scale well and the readwrite throughput and everything will be much higher, transaction throughput.

So Quorum is an example of a private enterprise blockchain, which is originally created by J.P. Morgan and it is still being developed by multiple different banks for their internal or bank to bank information sharing.

(Refer Slide Time: 25:07)



So given smart contracts as building blocks, you can build an application which consists of many smart contracts working together hand-in-hand. And these are called distributed applications or dapps. So eventually, you know when you are building let us say a medical record management system, which is based on blockchain or you are doing a land record management system using a blockchain then you have to have an interface with the real world.

And this interface with the real world is through web technology, or Java scripts and HTML, CSS and etc. So people will actually interact with the blockchain using web interfaces. And this web interface will hide the fact to the regular user who are going to do transactions for example by only interacting with the web interface. They will not know underlying whether it is a blockchain or it is a whether it is centralized database.

And when they click a button or fill in some form, some smart contracts will be invoked. And those smart contracts will do their work and then give you some output and you will see it on the web. And therefore, it will look like an application. But this application is distributed in the sense that all the things are happening on a distributed

set of nodes connected over a blockchain and therefore they are called distributed applications or dapps or dapps.

So the application that is hosted will be on a web server or on a decentralized file service such as Swarm or interplanetary file system or IPFS. So this is called the interplanetary. So look it up. So dapps based solutions are available for say record keeping, finance, supply chain, real estate, marketplaces for buying and selling products customer to customer without going through an Amazon like system, or even Amazon could actually do a dapps for in the future for their ecommerce.

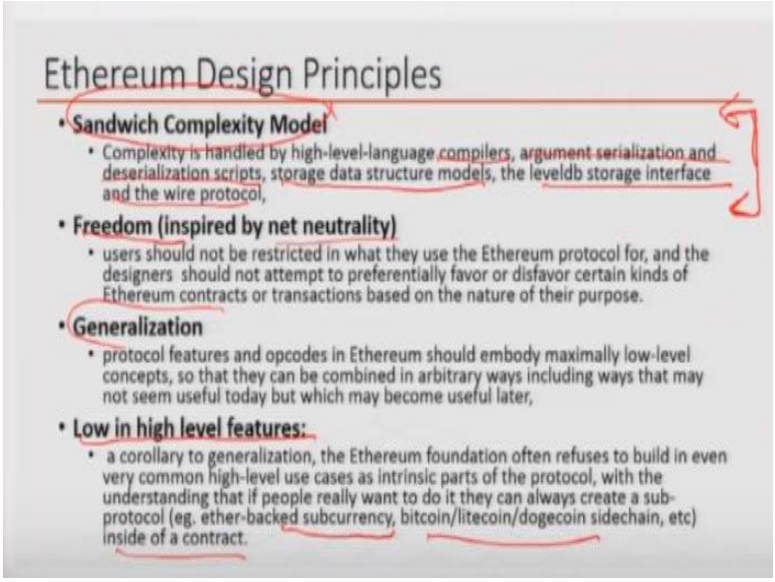
So that is where we will stop in this module. And then we will start in the next module.

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

Lecture - 15

Welcome back to the blockchain technology and applications.

(Refer Slide Time: 00:21)



We were talking about dapps or Ethereum based applications that are interacted with by users through web interface and the underlying technology of smart contracts and blockchain can be hidden. Now as I said, that will give you some indicators as to where to go and learn more about it towards the end of this lecture.

But before that, I want to point out a certain few other things about Ethereum which Ethereum foundation, so these are from the documents of white paper of Ethereum Foundation, which basically explains their choice of the design of Ethereum which is quite different from bitcoin, although both are currency, cryptocurrency block chains and both are permissionless blockchain and both are pseudo anonymous.

And both have the notion of transactions and mining and hash puzzles and everything. But still they have very different design principles on which the entire design and architecture of Ethereum was done. And this is done because the Vitalik Buterin who is the originator of Ethereum was using bitcoin blockchain for a while, and then he realized that there are certain things that are not very easy for with bitcoin.

And there are certain design principles that he did not like of bitcoin and therefore he is explaining here and I think you should know that, because one of the goal of this course is to actually give you more conceptual handle on the blockchain technology then nitty gritty programming practices of blockchain. So you should be able to explain to somebody who is asking what is the difference between Ethereum and bitcoin.

Of course, you can talk about smart contracts you can talk about Solidity which is Turing-complete language. But there are certain other things more higher level principles that are at play here. So Ethereum has this notion of what they call Sandwich Complexity Model. So sandwich basically means that it has three layers right. So top layer and bottom layer are bread and then middle layer has something like a meat or some vegetables or something.

So here also the idea is that all the complexity is in the top layer and the bottom layer and then the middle layer is kept as simple as possible. So complexity is handled at the high level with the language, Solidity language and the language compiler argument serialization and deserialization scripts, that is when you have complex data structures that are part of your arguments in a function called smart contract, how they are serialized because they have to be traversing the network, right.

So therefore, then there is need for serialization but the user does not have to do these things themselves. There are already scripts, which do the serialization, deserialization automatically. So user will use it like a normal function called storage data structure models. And then the leveldb storage interface at the bottom layer. That is where the information is stored.

And the wire protocol that is the peer to peer protocol that is used for interacting between the nodes in the blockchain. So that is the idea of Sandwich Complexity Model. So this is not something that the bitcoin used, right? So bitcoin has kind of a monolithic complexity and therefore the user, like who puts transactions have to know a lot more the working of the bitcoin mechanism in order to write complex scripts.

Here, the programming model is very much like a regular program, like when you write regular programs, you do not think of like the computer architecture, the registers and program stack and all that stuff when you write your program. Similarly, in Solidity when you write smart contracts, you do not think of the underlying architecture microarchitecture of the EVM and all that issues.

So that complexity is kind of hidden from you. So there is another idea of freedom or net neutrality. So if you have noticed in bitcoin the, it is very biased towards only bitcoin transaction. So we talked about how you can use bitcoin to actually store some data permanently or timestamp some data permanently, but usually those are not favored. So the way the bitcoin has been designed is kind of singularly meant for the bitcoin transactions.

So what they decided is that we should not restrict the users so that they have that the system or the platform preferentially disfavors a particular type of transactions or particular type of applications. So it is kind of net neutrality inspired idea that any kind of application it could be cryptocurrency application, it could be ecommerce application, it could be just data storage.

Like land record, where you are trying to guarantee the integrity of the data, or it could be a grade management system. All that stuff can be actually managed in this with equal difficulty or equal simplicity. So that is the freedom principle. Generalizing principle is that all the protocol features, and opcodes in Ethereum, are actually low level concepts. So they do not create high level opcodes, which do very high level that an opcode that basically it says, do something very complex.

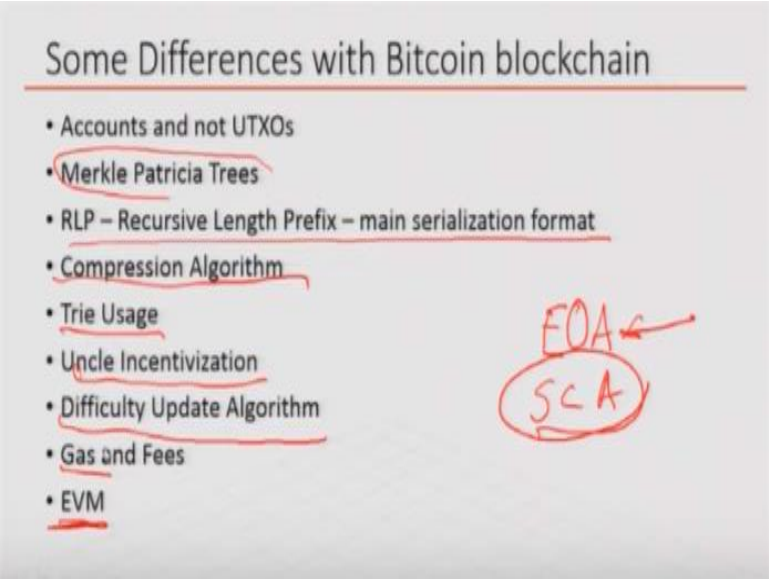
So everything is of very low level, simple opcodes semantics are very simple low level ideas. So one can actually use them in arbitrary ways. So create very complex functionality. So more it is like, you know like, if you want to do something very complex, assembly level programming might be more flexible for you. Then let us say you are using some high level programming language like a functional language.

And then you have to, you know, do a lot of trickery to make it do something complex and something that is not meant for by the language designers. Here, they did not do that choice and it is very generalizable. And therefore, as a principle, as a corollary to this principle of generalization, they have very few or almost no high level features. And they expect that if you want to create high level features, you can create protocols on top of Ethereum protocol.

So you can make ether-backed subcurrency, you can define your own tokens. And like this is how the ECOS are done. And you can also make other kinds of coins and etc., using sub contracts. So that flexibility is there. So four principles that design principles is that a complexity is kept, you know out of the users purview. The freedom in the sense that it does not disfavor a particular type of production or favor a particular type of application on top of Ethereum.

It is generalizable and generalizability is probably the core to this freedom and because it is basically uses all opcode semantics are very low level concept based and not high level and then therefore the as a corollary it is low in high level features. So remember, this very high level design principles that drove the creation of Ethereum and the design and architecture of Ethereum.

(Refer Slide Time: 08:48)



So another difference that is interesting in case of you know difference between bitcoin blockchain and Ethereum is in bitcoin there is no concept of real account. Everything is kept track of by unspent transactions, right. So whenever you give some

money to another person, you have your digital signature on it. And then whenever you want to use that money, you have to use that as an input transaction.

And then you create output transactions. So these UTXOs are the ones which are not yet used. So this UTXOs are kept track of. So that is kind of the state of the current state of the blockchain. So in Ethereum, however, we do not keep track of UTXOs, we have the notion of accounts. So every account the accounts are two types. One is called EOS or external user accounts.

And another is the smart contract accounts, right. So externally owned accounts and smart contract accounts. So these accounts have their addresses, they have ether balances. The balance could be zero, or the balance could be positive, but they have balances. So when you send money to somebody, let us say a real person with an externally owned account, then that balance will go up.

If the person spends money from its balance, then the balance will go down. Similarly, the subcontract accounts also can keep ether as balance and can send ether to externally on the accounts or to other smart contracts for getting things done. So therefore, they are the notion of accounts and not UTXOs. In Ethereum, we saw that the transactions are kept as Merkle trees. In here, we see a little different concept.

They use something called Merkle Patricia trees and we will show you what this means. They have a different serialization format called Recursive Length Prefix. And they have a compression algorithm. They use tries, which is a particular data structure on which the Patricia trees are based.

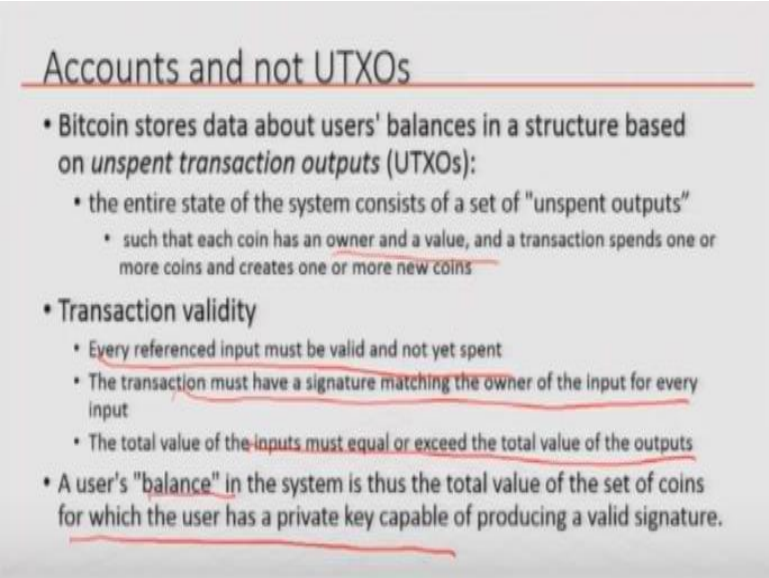
And they have this notion of Uncle incentivization which basically means that you know whenever so your parents in case of bitcoin blockchain, but there is no notion of uncle because uncle is actually a branch that has been abandoned after the race condition that happens when multiple branches starts coming up. But eventually everybody starts building on top of the longest branch.

So the other blocks that would be called uncle or grand uncle, they now have no meaning. But here we have the incentivization for uncles because then if two of you

are making the next block, eventually one of you will become uncle and then there would be some ether that you will get for becoming uncle. So Difficulty Update Algorithm that bitcoin also has but it is a different algorithm than bitcoin.

Bitcoin also changes the difficulty of the hash puzzle. This one also does that. And the notion of gas is completely not there in bitcoin. Bitcoin has notion of transaction fees. Here we pay gas. And because we pay gas, we always have to pay fees. And then the Ethereum has this notion of EVM which runs bytecode and which runs Turing-complete bytecode which of course, is one of the major differences.

(Refer Slide Time: 12:42)



So something to be said about this notion of accounts versus UTXOs. So bitcoin stores user's balances in a structure based on unspent transaction outputs or UTXOs. So the state of the system is a set of unspent outputs. So whenever you try to make a transaction, the input transaction should come from the UTXOs, right. Because then you know that there is no double spending attempt. So each coin has a particular owner.

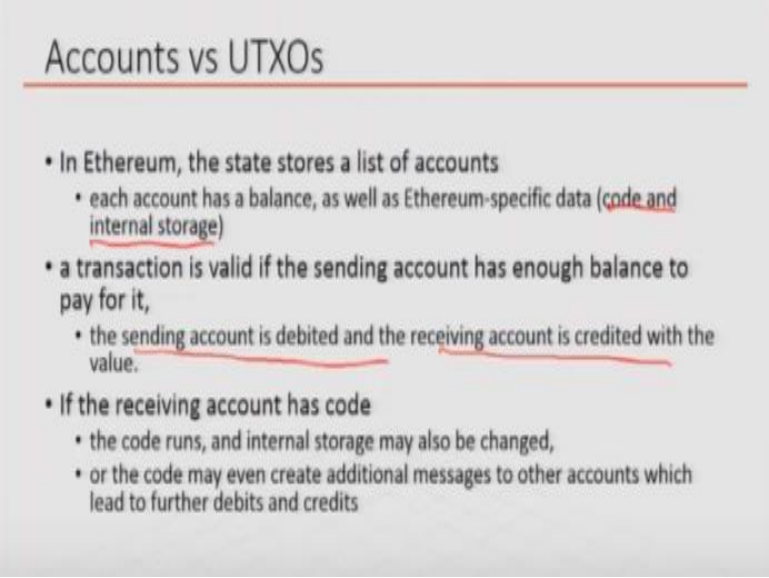
And as we know that in bitcoin, when you create a coin, when you either mine a coin, you become the owner, and that mining transaction is associated with the, your ownership, right? So when you want to spend that coin, you have to use that transaction in which you mined the coin as an input transaction. And then you have to, if you are giving the money to somebody, there will be output transactions.

And you have to give the balance to also a different address as we know because otherwise there will be replay attacks. So each coin therefore has a owner and a value and transaction spends one or more coins and creates one or more new coins in bitcoin. And the idea is that every referenced input must be valid and not yet spent.

So it should be from the UTXO. And the transaction must have a signature matching the owner for every input and total value of input must be equal or exceed the total value of outputs. If the inputs exceed the total value of outputs, the difference between input and output total will be lost forever. The user's balance is the total value of the set of coins for which user has a private key capable of producing valid signature.

So the balance is based on the UTXOs for which it has a valid private key. So that is the bitcoin way.

(Refer Slide Time: 14:49)



In Ethereum, the state stores a list of accounts. And at any point, the current state has a list of accounts which are active and they have balance. So this data that is associated with the account is balance and the code if it is a smart contract and internal storage, right. So various kind of other information, the program state basically because you when you execute the program, you may change some state variables of the program.

So there will be those as well. Now a transaction is valid if the sending account has enough balance to pay for it. So it will what it will check is that for the account, it will

check whether the balance is enough. It does not care whether that money came through another which transaction say. So in case of bitcoin, you identify exactly which transaction gave you that money. That becomes your input.

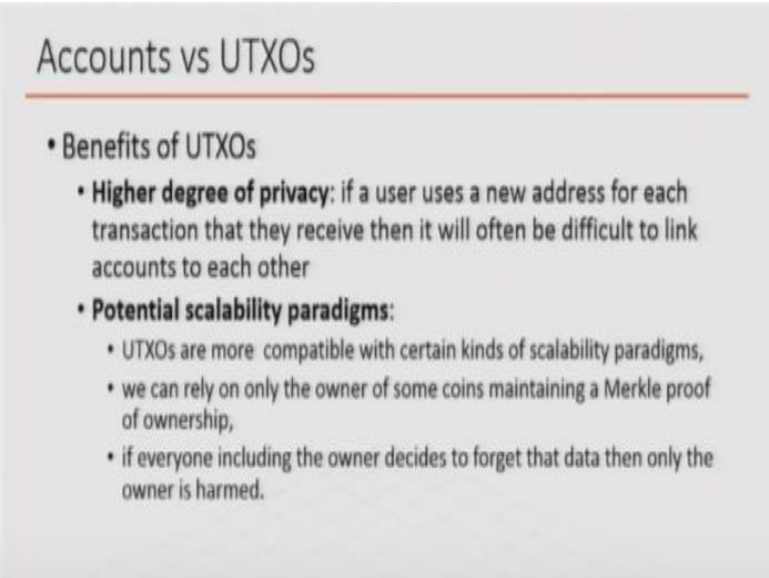
And when you create an output transaction giving money to somebody else, then you are creating a new coin that you are giving to somebody else and that somebody else then we will have that unspent transaction until it spends it. Now in this case, I might have multiple transactions, let us say one for five ether, one for two ether, one for one ether, then I will have an 8 ether balance.

And this money becomes these coins are fungible in the sense that they are replaceable with each other. I do not know which of these eight coins came from the first transaction which came from the second transaction, which came from the third transaction. The only thing that happens is the sending account is debited and the receiving account is credited with the value.

If the receiving account has a code, then it might do something else after it receives the ether, depending on what function was called to send the ether and it may run the code and there may be some state change also. For example, it might say that some flag may become true to false or false to true or it might actually have some other information that changes that can happen.

And even these there may be a chain of transactions that may be created. Because once this smart contract gets your ether, it might actually create other transactions to other contracts, send some part of the money to another account all kinds of chain series of transaction will may also ensue when you do that.

(Refer Slide Time: 17:28)



So benefit of UTXO is that it gives you a higher degree of privacy. So because remember in UTXO, every time you let us say you have 25 bitcoin, and you are giving somebody 5 bitcoin, but you give the 20 bitcoins to a new address. And then when you want to spend that 20 bitcoin, then you have to use that transaction with the new addresses private key to unlock the 20 coins.

And you will let us say you give another 5 to somebody else and then a 50 and you give to a new address. So addresses proliferate and therefore, when somebody collects all the data from the past transactions blocks and try to figure out who is who and all that stuff, it becomes more difficult because same guy will have many addresses. Whereas, if you have an account, account has a given address.

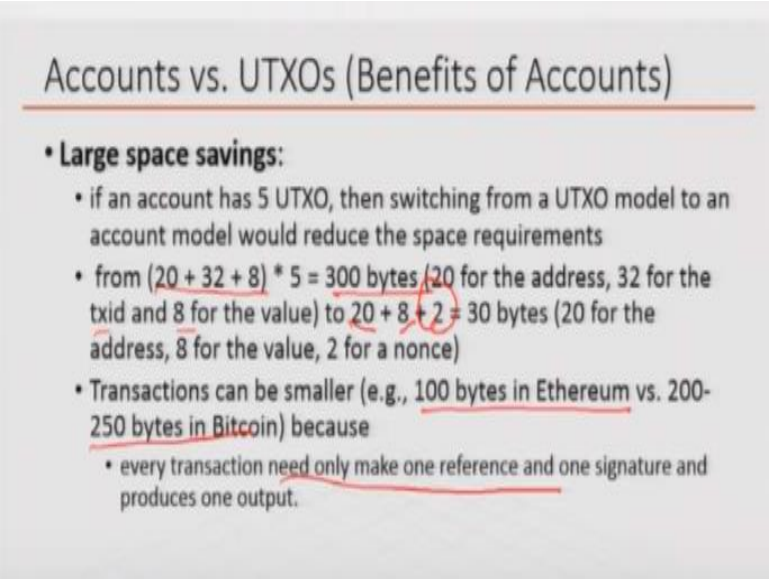
So therefore, and then you make a transaction to you start with 25 ether, you give 5 to somebody, then you give another 5 to somebody. So all those things will be easily tied to you. However, obviously the account is now, account address is actually a public key in some hash form. Therefore, that does not mean that you will be completely you know deanonymized.

But in bitcoin, it is even harder to put together all this transaction to attribute all these activities to you. And also UTXOs are more compatible with certain kind of scalability. And the owner of the coin maintains the Merkle proof of the ownership which basically means that it has the signature, it has the ability to do the signature

and it also has the transaction history, the hash of the root of the transaction history, which is called a Merkle proof of ownership.

And if the owner decides to forget that data, then only the owner is harmed. In case of an account, it is if you forget, then anybody who is supposed to get some activities or everybody who has the ability to make transactions on that smart contract in which the ether balance is there, then they cannot none of them can use the ether if that is forgotten.

(Refer Slide Time: 20:00)



However, for accounts there is a lot of space saving. So if suppose your wallet has 5 UTXOs associated with it. I mean you only know because you have your wallet has all the corresponding private keys. So you know the other guys may not know that they all these UTXOs belong to you because they have different public key associated with them.

But you as an account owner would know that I have this 5 UTXOs because I have this 5 keys and they are unspent. So however to keep that information on the blockchain you need 300 bytes 20 for an address 32 for the transaction ID and 8 for the value. So these are bytes and that times 5 for each of the UTXO you need 60. So 60 times 5, 300.

Whereas in case of an account, you do not have to make it 5 times, right, because it is a single account. So you need to know the address. You need to know the value. And

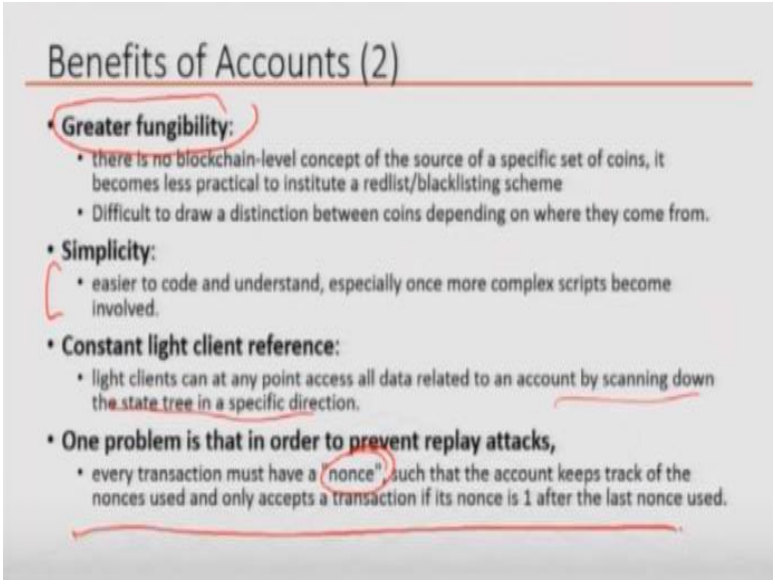
you do not need to remember the transaction ID that brought that value in. And then you need 2 bytes for a nonce. Now why is nonce there? Because the problem with the UTXOs are used because we want to stop replay attacks, right?

So we actually, for every transaction we create new addresses for the balance and therefore replay attacks cannot work. But in case of accounts, you have to use some kind of a way to distinguish every transactions. So there is a notion of a nonce. Nonce is a once only number. So it is a it may be initially a random number. So this actually is incremented and therefore, you cannot replay it.

So transactions can also be smaller, so 100 bytes for Ethereum where 200 to 250 bytes in bitcoin. And because every transaction need to make only one reference and one signature, whereas in case of bitcoin every transaction requests all the let us say you want to spend 25 bitcoins. One single UTXO which has given you 25. Let us say you have 5 of them. So 5 UTXOs will be your input and each of them will have their own transaction ID and address and the amount and digital signature.

So it will require a lot more information per transaction. Whereas in this case, in Ethereum, you only have to say the account address, the nonce and therefore you will need less amount of bits to formulate the transaction.

(Refer Slide Time: 22:58)



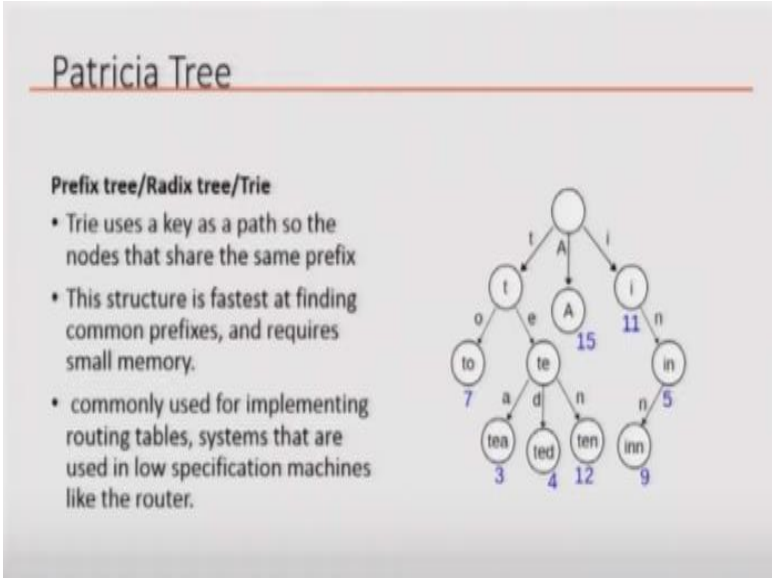
So the benefits of accounts is greater fungibility, which we already talked about. That is all the different sources are all put together. So you do not have to remember that

this came from this and this came from that. So you cannot blacklist or red list a coin. You cannot say that this coin came through wrong type of sender.

So this UTXO now get banned. You cannot do anything like that, because all these things are fungible unless you want to ban the entire account, you cannot really do anything. So coins are not distinguishable which transaction they came from. Of course, accounts are much simpler to code and understand because you know the notion of accounts from your banking and other kinds of activities.

And the light clients at any point can access all data related to an account by scanning down the state tree in a specific direction. And the replay attacks are prevented through the use of nonce. So we already talked about that each transaction is associated with a nonce. So that basically takes us to the end of the discussion on accounts.

(Refer Slide Time: 24:13)



So when we come back, we will talk about the some of the data structures used in Ethereum, as well as we will talk about the, you know give you some pointers as to how you can start making yourself more familiar with making, creating smart contracts and Solidity language and so on.

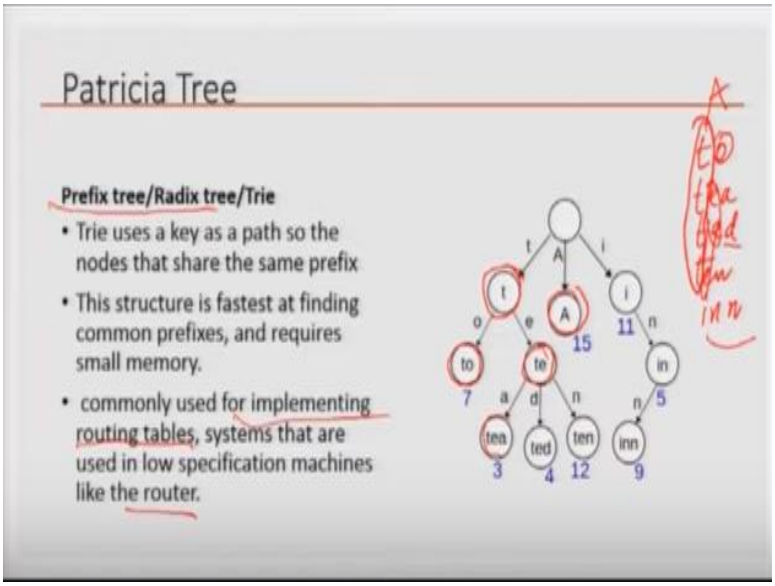
Introduction to Blockchain Technology & Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

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, 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. 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 $\log_2(100)$, which would be about height of let us say, $\log_2(100)$ 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.

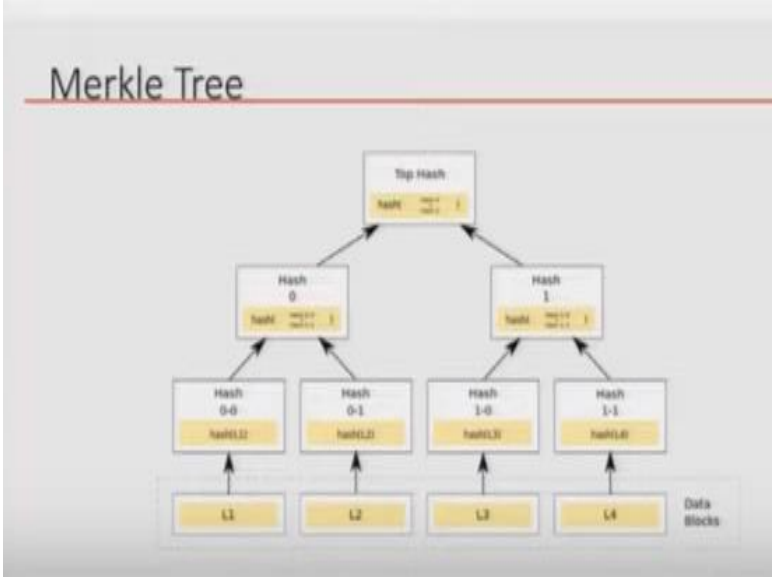
So in this case, what you do is that we see that there are four of the strings starts with the same prefix, so t. So I will create a root and then I will have a child which is marked whose paths is marked with t and then I will have note for t and then I see that after t there are two with e and one 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

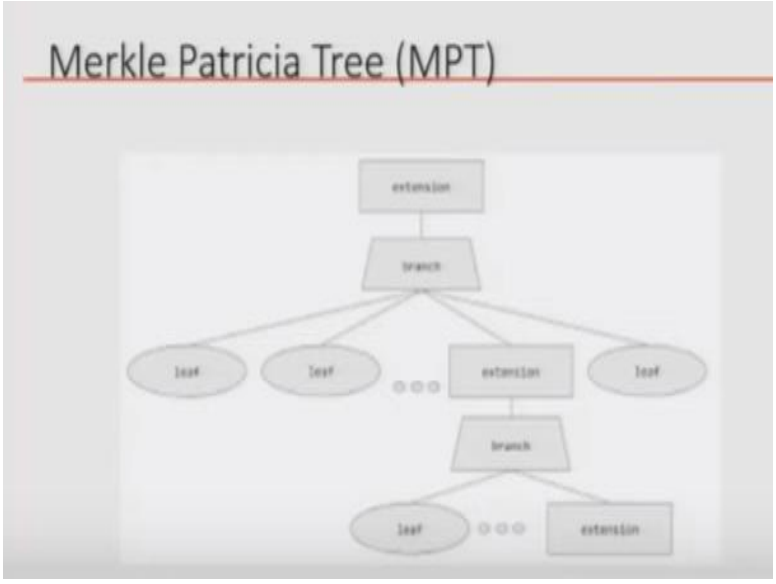
for implementing for example, routing table and routers and so on. So this is not something new that is being invented here. But they decided that this is how they are going to store their account informations.

(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.

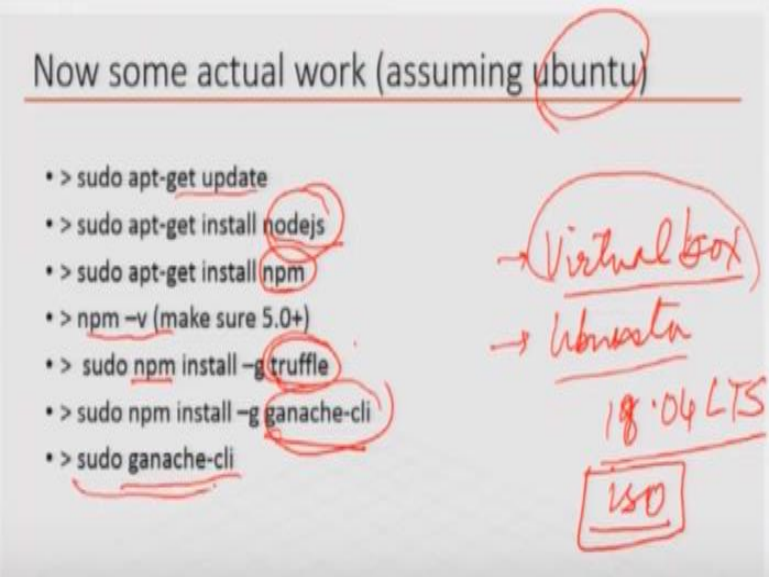
(Refer Slide Time: 05:11)



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 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 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 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.

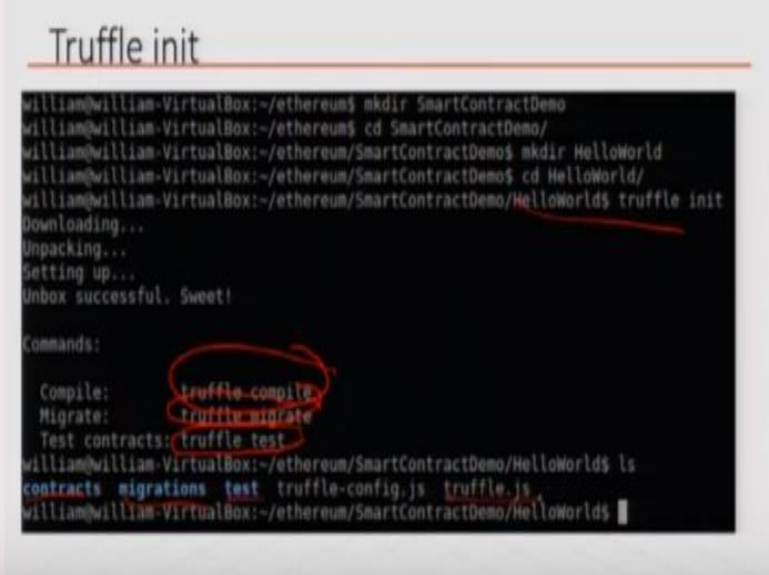
But this gas and gas price and all that stuff is going to be fake because this is not against the Mainnet so you do not really have to pay any money and then it will start listening for nodes to do things at this port 8545. So this port so when you deploy a transaction, it has to refer to this host and 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)

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.

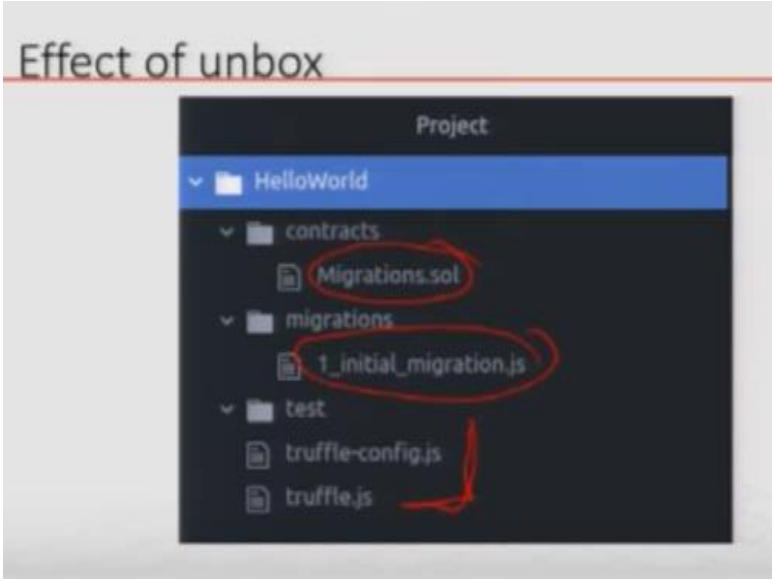
(Refer Slide Time: 10:58)



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.

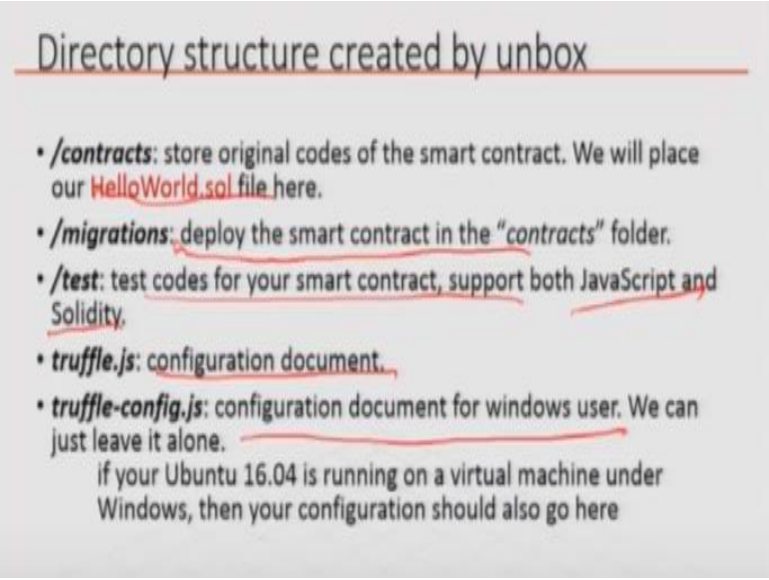
Because when you create a smart contract, you also have to create JavaScript or whatever to actually deploy the contract or call functions on the contract and so on. There is a test directory and there will be configuration JavaScript files. And now you have to start working with this.

(Refer Slide Time: 11:50)



So if you look at this thing, then you will see that you have HelloWorld and the HelloWorld 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)



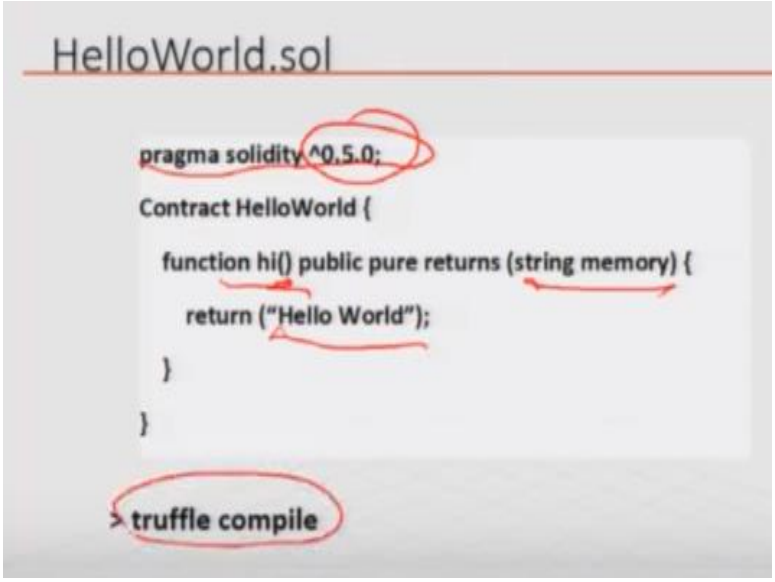
So contracts is where you store the Solidity code of the smart contracts. Now we will create a HelloWorld.sol file, and put in the contracts and fill it up. Migrations is also another smart contract that is going to be used to deploy this contract. And test is a test code and it can support both JavaScript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

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)



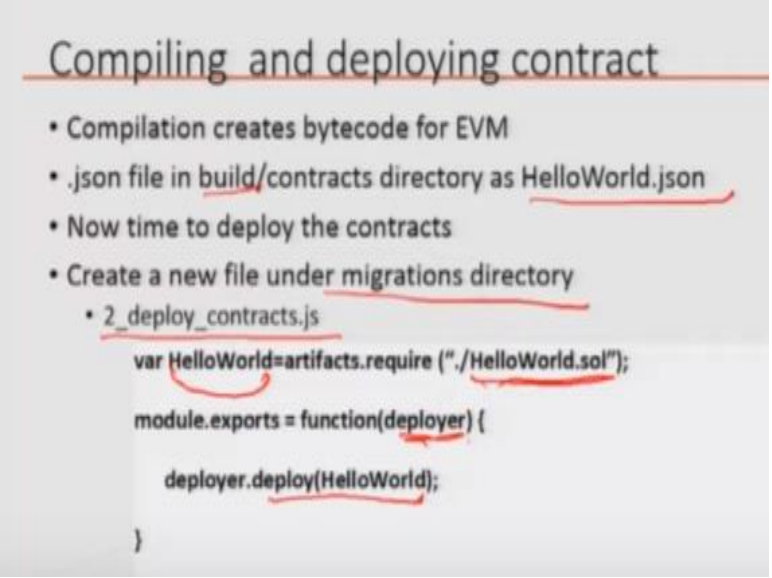
So this will look something like this. So pragma, solidity this says what is the version of Solidity. Solidity is a evolving language in last 4, 5 years. So right now I think it is 0.7. So currently, so this is from last year. So most probably you will see something like 0.7 or something here. And then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



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)

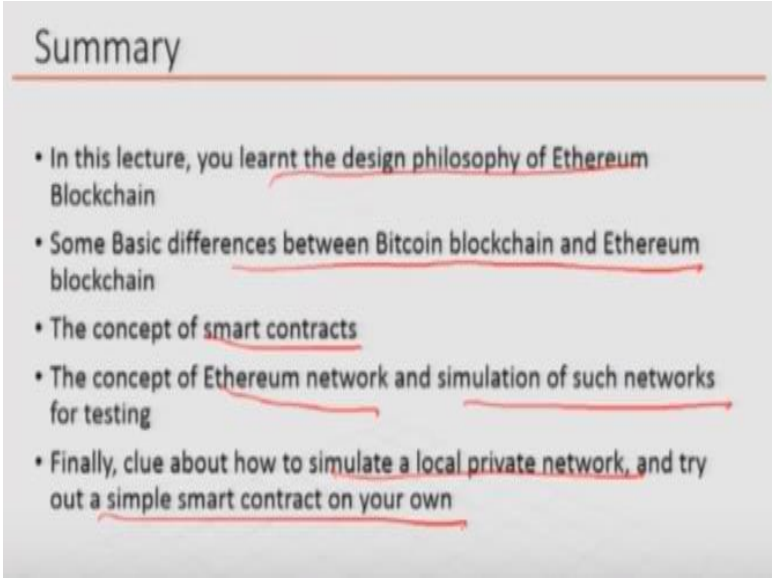
So as I said that in the truffle.js, you have to make this uncomment. So mostly they will come commented like this. So you have to basically uncomment the relevant part, so that you know that your host on which the network host is local host, and then the port at which it is listening is this. And then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

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)



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.

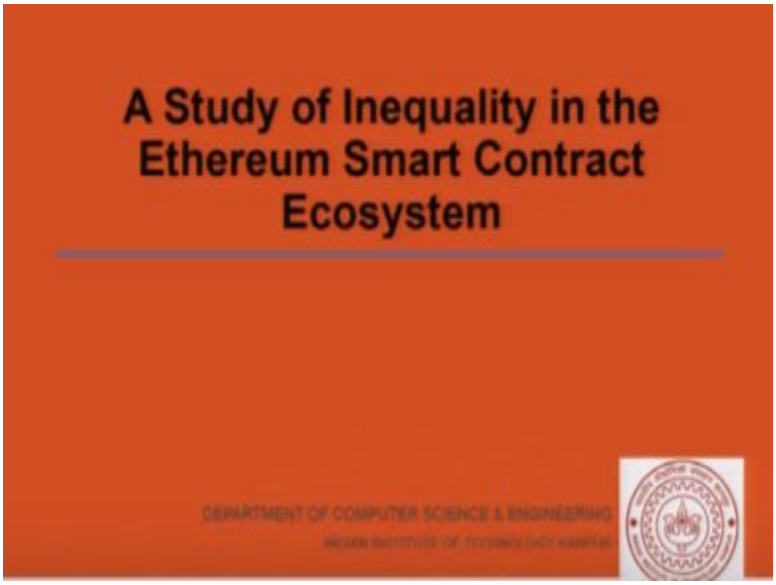
The concept of smart contracts, the concept of Ethereum network and simulation of such network for testing, and finally, a clue about how to simulate a local private network and try out a simple smart contract on your own. So in this course, so we are about four weeks away from the end, and we have to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

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

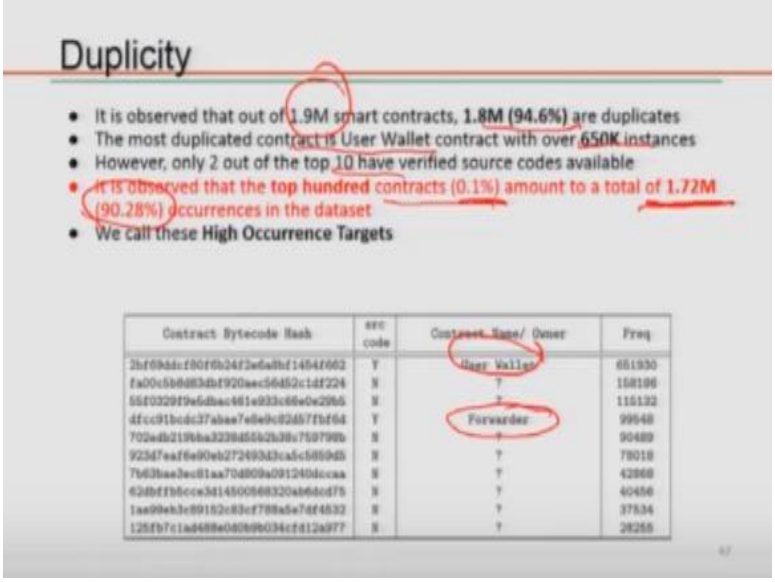
(Refer Slide Time: 22:03)

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 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)

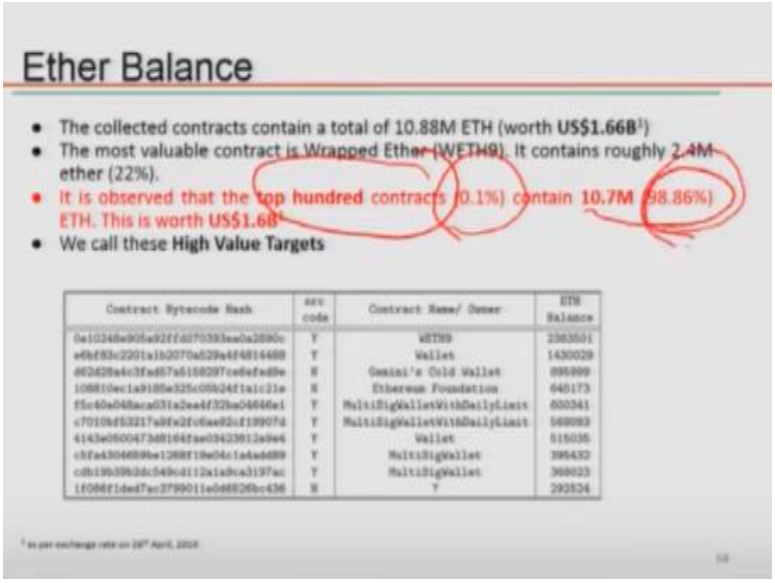


So what I am going to show you is that see it is observed that out of 1.9 million smart contracts, 94.6 that is 95% are duplicates. So about 0.1 million that is about 100,000 or so smart contracts are actually original smart contracts. Most duplicated contract is of course, the user wallet contract,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

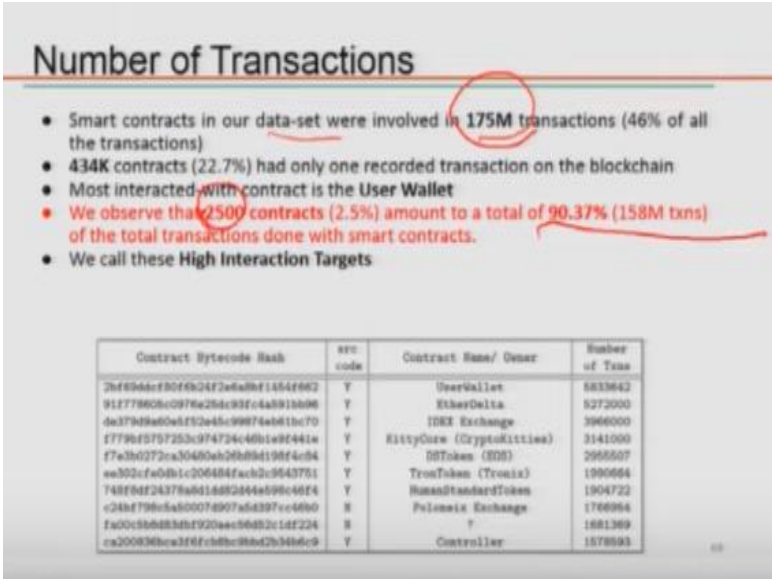
(Refer Slide Time: 24:30)



And then we 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)



And we saw that smart contracts 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 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.

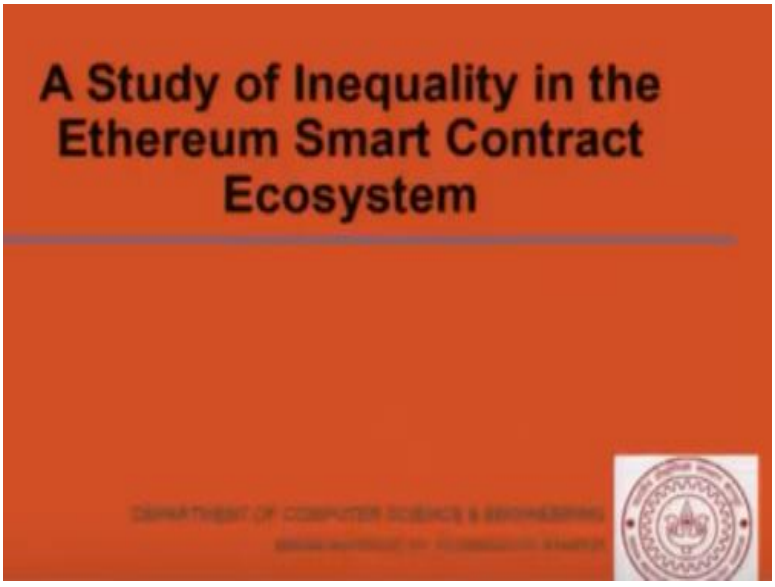
Introduction to Blockchain Technology & Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

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.

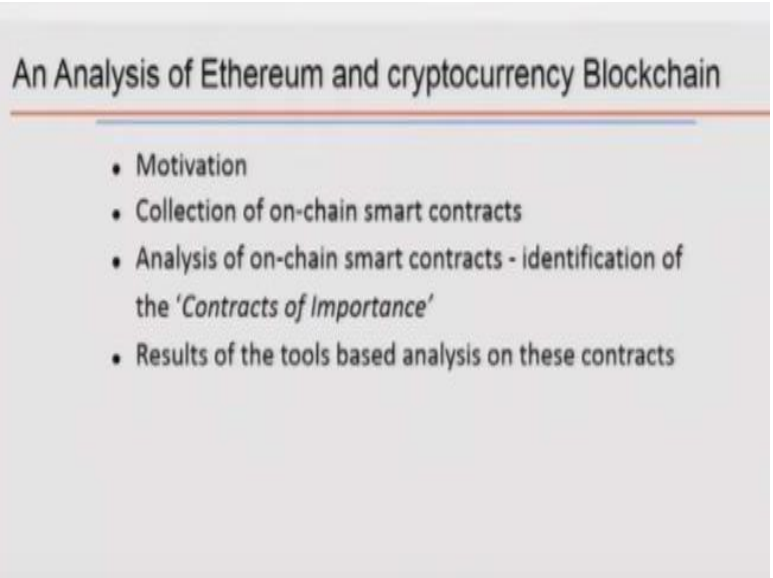
So today what I want to do is that we want to study some of the stuff that is quite concerning about cryptocurrency blockchains. And so that you have a very good idea about the risks and some of the pitfalls of this kind of permissionless cryptocurrency blockchains.

(Refer Slide Time: 01:14)



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)



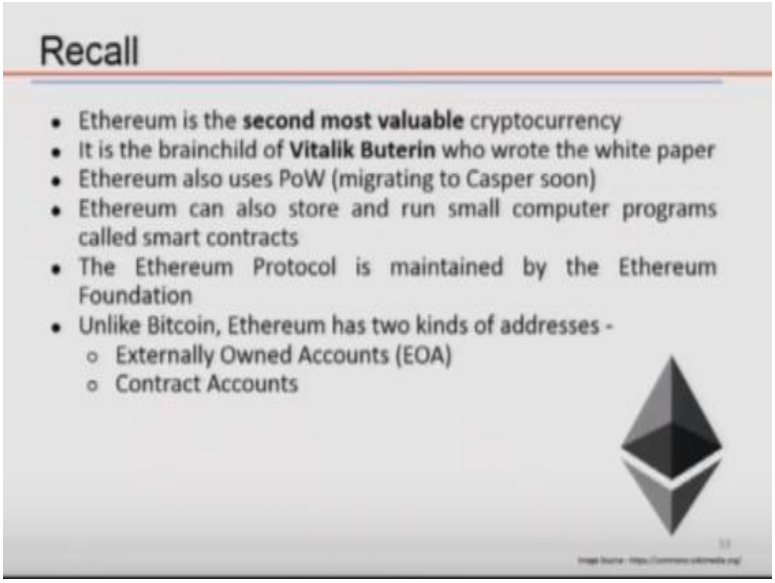
So what we want to do is that we want to first motivate why we are studying this. Second thing that we want to do is that, we want to tell you how we collected all the smart contracts that were available until last year April. And then we want to talk about how we analyze them and what kind of results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



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.

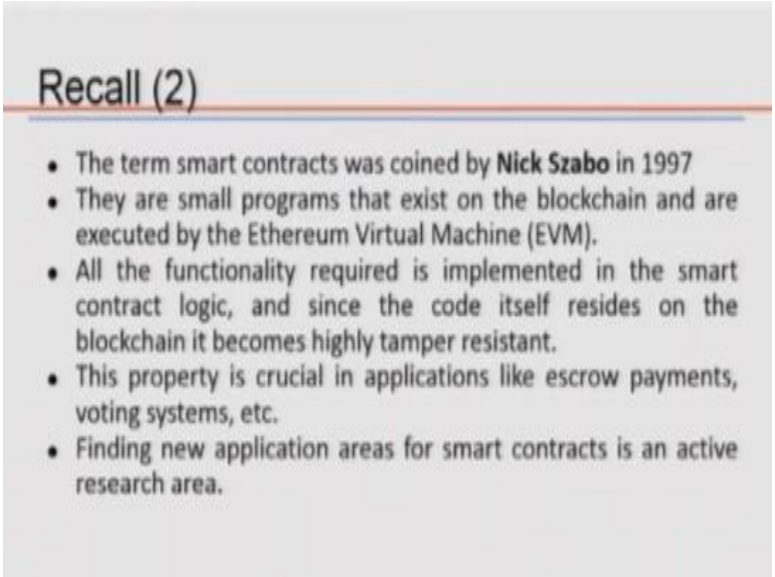
And these programs are universal programs, they are Turing-complete, and therefore all kinds of computation can be done, and therefore, the appeal of Ethereum in applications outside cryptocurrency is very high. And the Ethereum protocol is maintained by Ethereum Foundation. As we will discuss later,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



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.

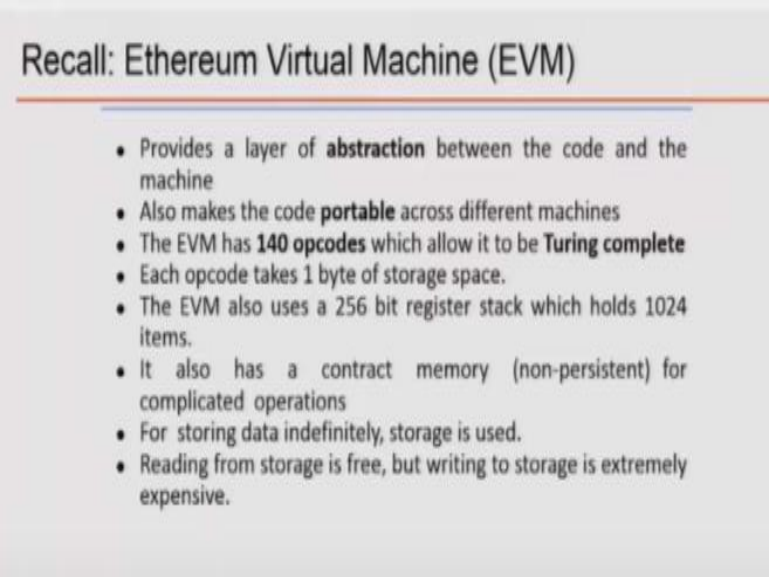
And the consensus mechanism works so that once a block is mined, then all the steps that were executed for all the smart contracts in the block that has been accepted and percolate in every node that is maintaining the blockchain has to also execute those. And the functionality is implemented in the 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)



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.

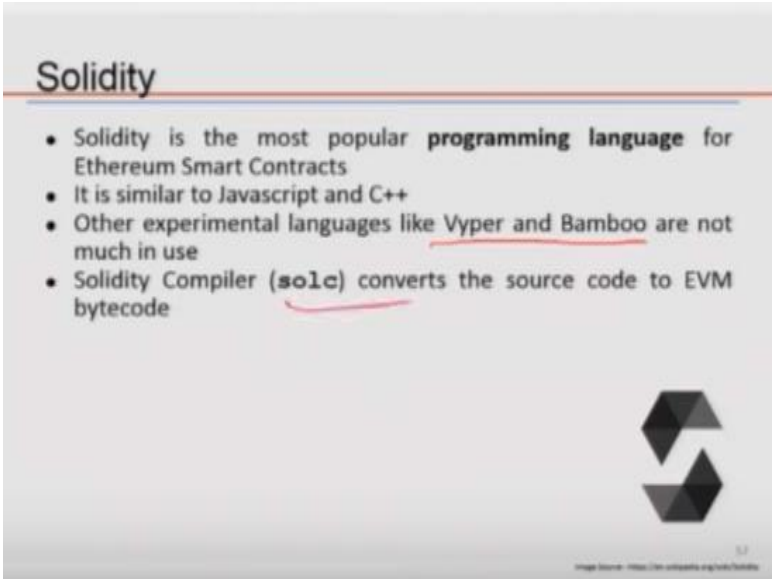
So any high level action that needs to be taken will be a combination of sequence proper sequencing of this low level opcodes to obtain a high level functionality. And each opcode is 1 byte long. And as we know that EVM has a stack, registered stack based execution model with 1024 items, and each of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

And it has contract memory, which is non-persistent, which 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)

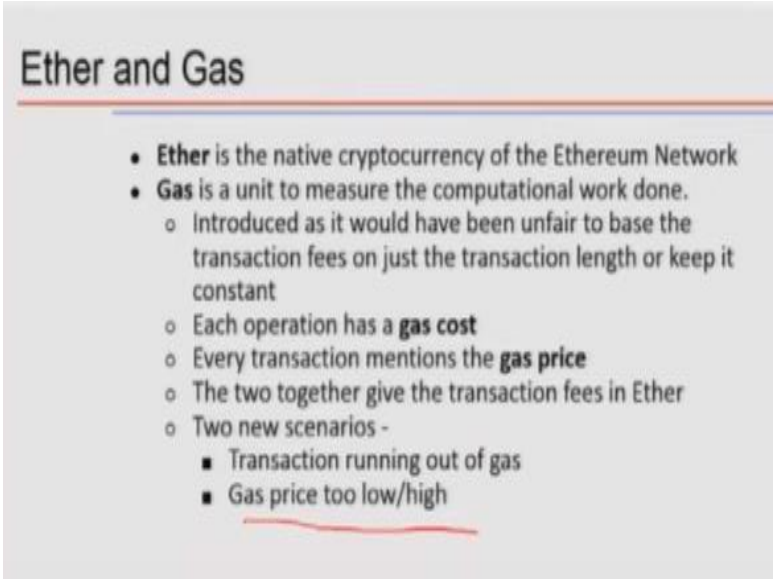
So as we know that, the source code is written in language like Solidity, and the Solidity is compiled into the bytecode. And that bytecode contains the smart contract creation code and the actual code that will be translation of the various functions in the contract and its storage information. So when 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)



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 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 taking unfair advantage plus possibly to get their transaction into the next block.

(Refer Slide Time: 11:46)

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 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 permissionless 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.

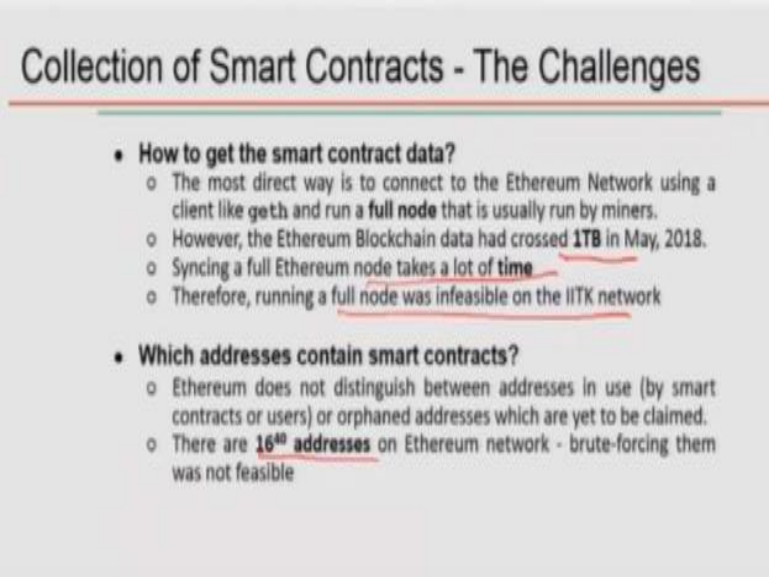
And second thing we want to analyze was there has been a lot of cyber-attacks on the Ethereum blockchain and other blockchains as well, including bitcoin, but in Ethereum, because of smart contracts, which are programs, so there are bugs in such

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)



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.

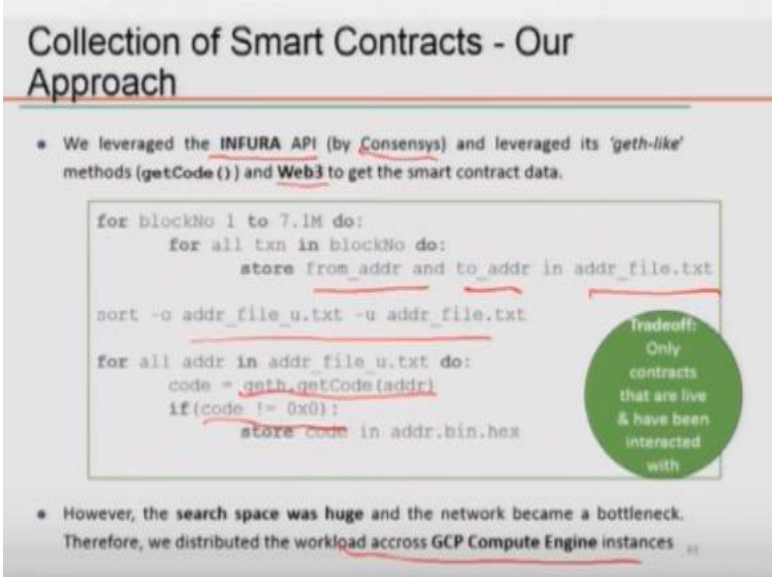
So if you want to run a full node, then you have to download locally, the entire blockchain and if you do that, then I have a local copy of all the blocks and transactions and the smart contracts. And I can do analysis. But Ethereum blockchain

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^{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 correspond to smart contracts.

(Refer Slide Time: 16:02)



So we use 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.

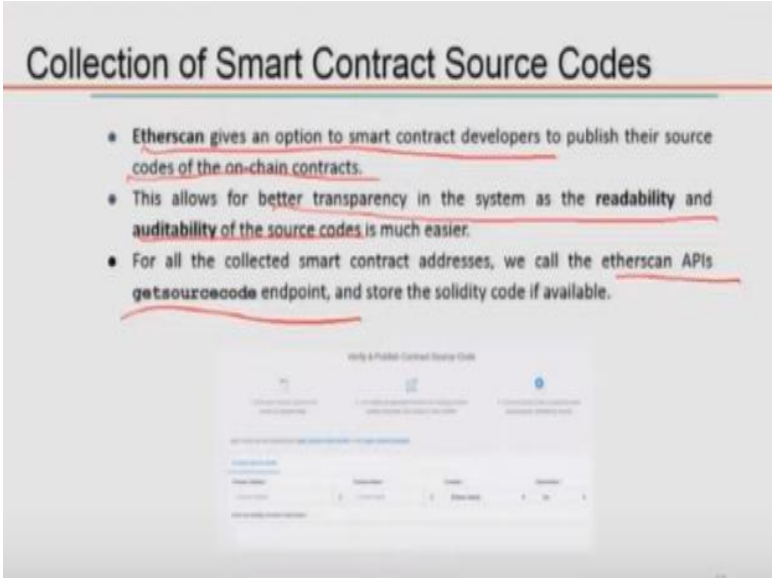
And if there is a code associated with that address, then we got the code, right. So that is what we did. So basic idea is that you collect all the possible addresses that are currently either appearing as a to address and or a from address. And then go through

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)

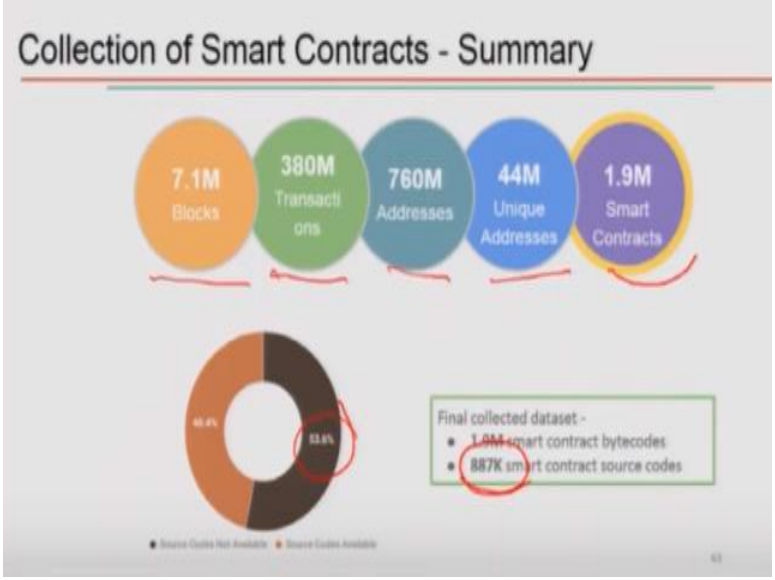


Now for some of the tools that we are going to use for analysis of the bugs. We also need the source code because not all tools work on the bytecode directly if someone requires the source code. So the question is what we downloaded like this, the process that I just described, was actually giving me 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)



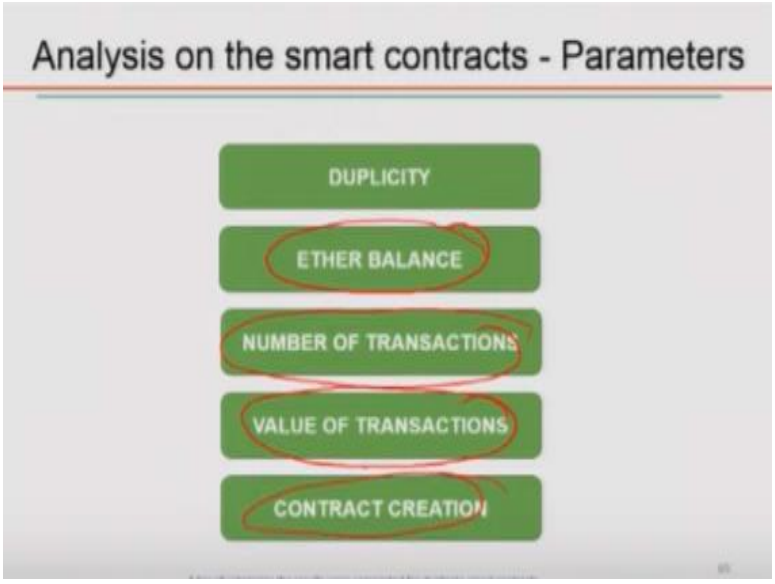
So we got at that time, so at the time when we stopped, I do not remember the exact date, we had 7.1 million blocks, 380 million transactions, 760 million addresses, 44 million unique addresses because of to and fro will have overlap. And then we found 1.9 million smart contracts. That is about 2 million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

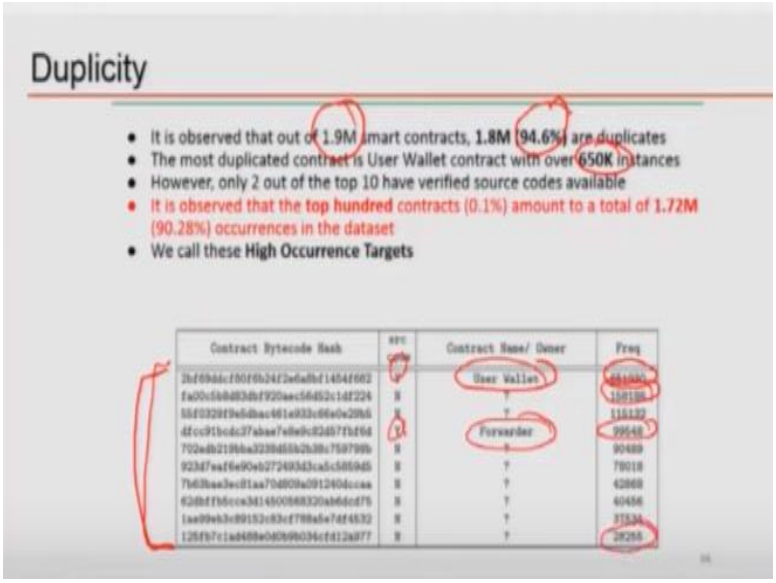
So how do you do that? What kind of analysis do you want to do to bust the myth that this blockchain cryptocurrency is highly democratizing?

(Refer Slide Time: 20:13)



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)



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.

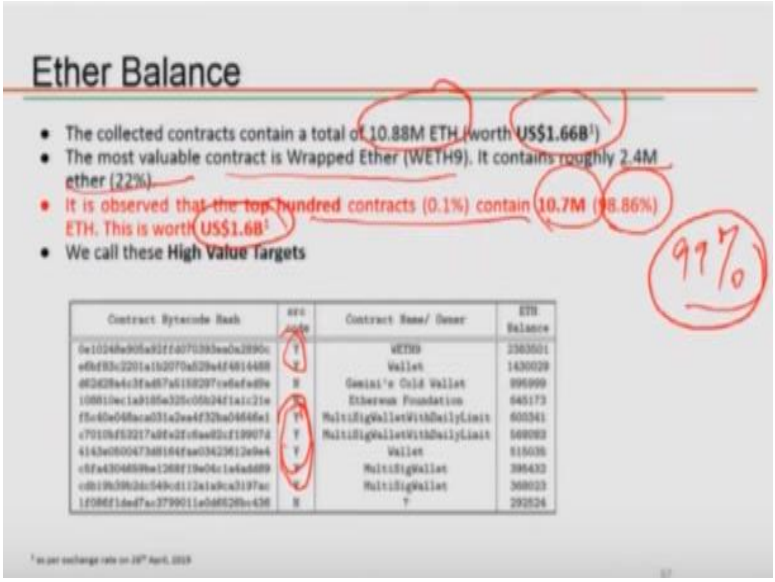
And then the most commonly duplicated one was the user wallet. So here is a list of the bytecode hash for the top 10 duplicated contracts. And on the top, you see the first one that is most duplicated was 651,000 times about 652,000 times. And that is the user wallet. How do we know it is user wallet 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 , 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)



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 cyber-

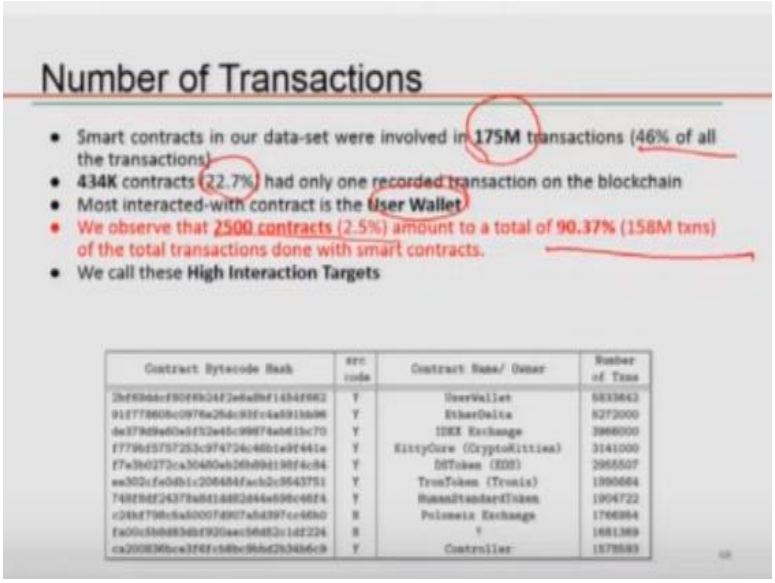
attacker wants to target a particular contract, and if that contract has bugs, then that will certainly be tried to be breached in order to make money.

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 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 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)



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.

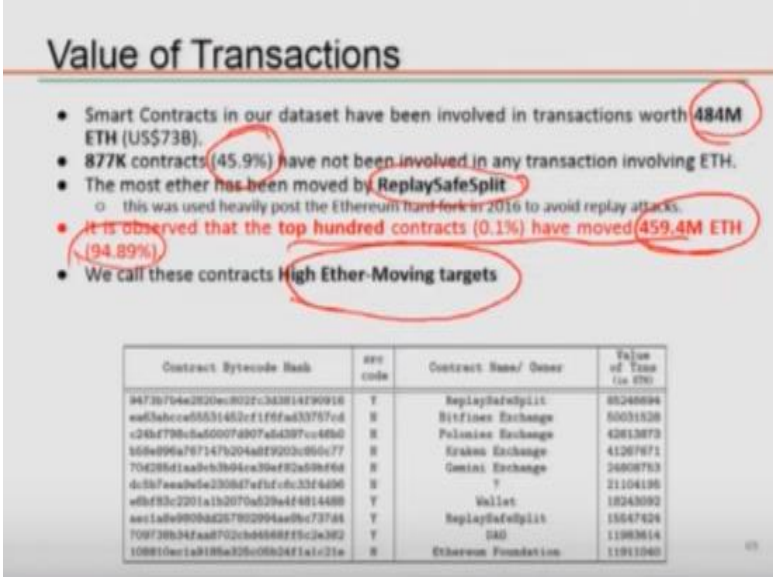
And we found that 2500 contracts that is only 2.5% amount to a total of 90.37 that is almost 91% of the total transaction. So again, we see a concentration of wealth and we see concentration of participation among this very small number of participants in the network. So when we come back, we will talk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

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 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.

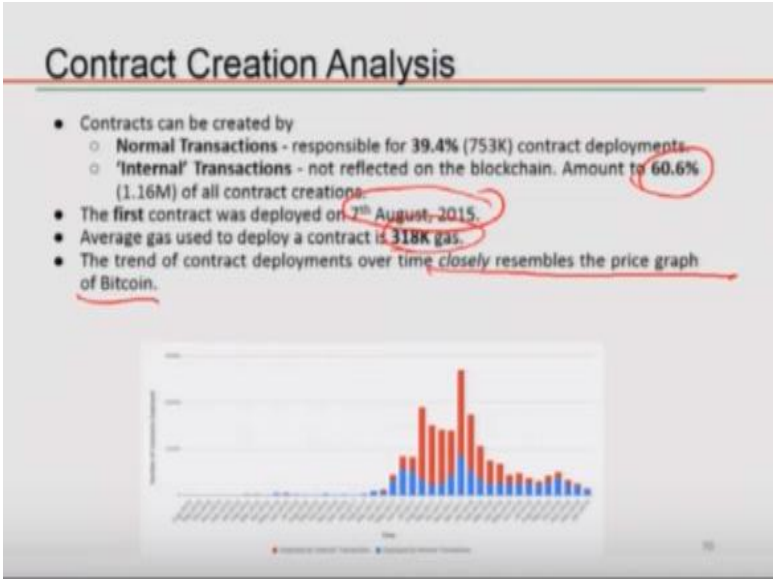
As we know that hard forking happen when you put extra restrictions on the transactions, and therefore the nodes that do not catch up will not ever get their blocks or transactions and blocks accepted. So that is how the hard fork happen. And obviously, the lot of ether was now sent to the new branch.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



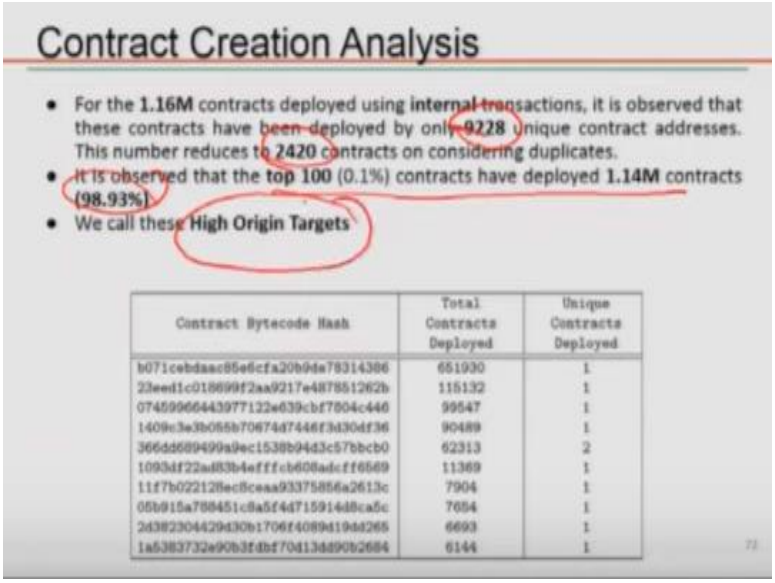
So contract creation analysis. So contracts are created by either normal transactions or by internal transactions. Normal transactions can be found on the blockchain in the blocks. But internal transactions, internal transactions are transactions, usually between smart contract to smart contract. And 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 kid 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)

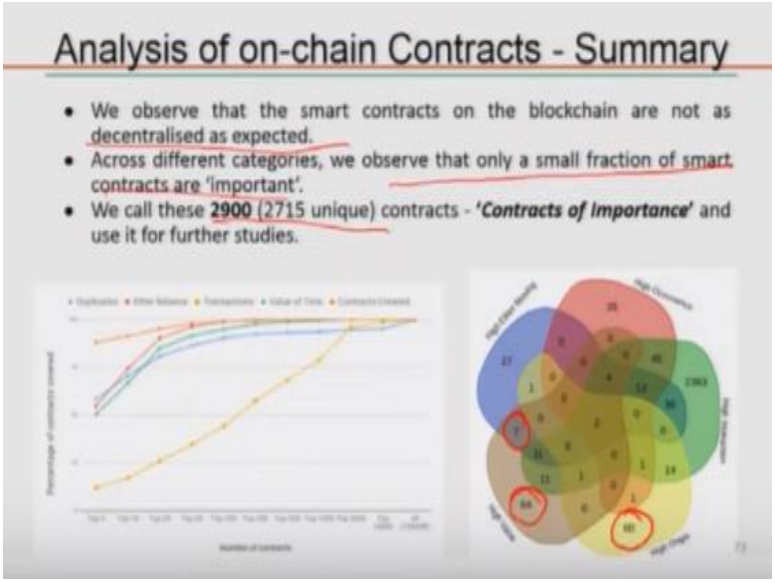


Contract Bytecode Hash	Total Contracts Deployed	Unique Contracts Deployed
b071cebdaac85e6cfa20b9de78314386	651930	1
23eed1c018699f2aa9217e4878512 62b	115132	1
0745996643977122e639cbf7804c446	99547	1
1409c3e3b055b70674d7446f3d304f36	90489	1
366dd689499a9ec1538b94d3c57bbcb0	62313	2
1093df22ad83b4efffcb608adcff6569	11369	1
11f7b022128ec8ceaa93375856a2613c	7904	1
05b915a788451c8a5f4d7159148bca5c	7654	1
2d382304429d30b1706f4089d19dd265	6693	1
1a5383732e90b3fdbf70d13dd90b2684	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 12-13 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 9228 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.

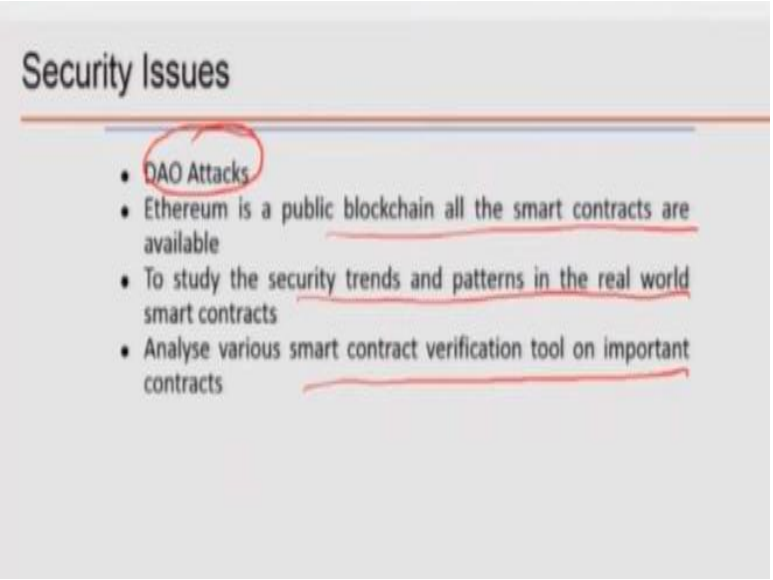
Not so, you know, conflict resistant, hash functions and cryptography, proper cryptography and so on, then I can say that the data that goes into the blockchain cannot be easily tampered. So it is kind of tamper resistant. So which gives me data integrity. But security is confidentiality, security is availability and security is integrity. 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 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.

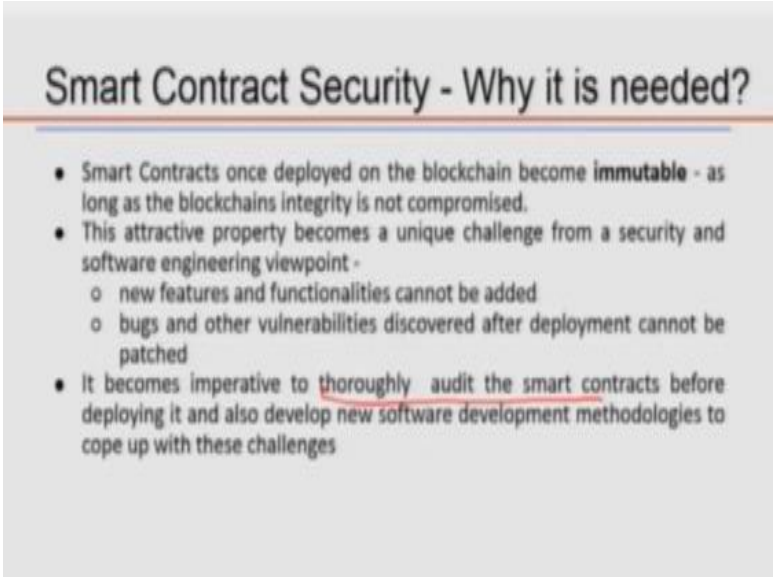
(Refer Slide Time: 12:56)



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)



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 a 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.

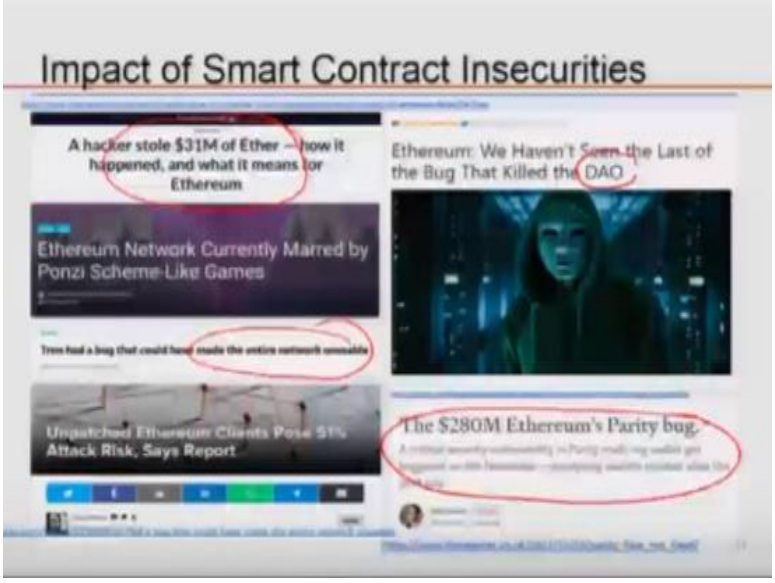
The code will be still there, but it has already changed state to a self-destructed state. Earlier it was called a suicide function. Now it has the suicide function has been deprecated. So in Solidity there, you can create selfdestruct function. And that is the

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 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 are 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 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 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 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, 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

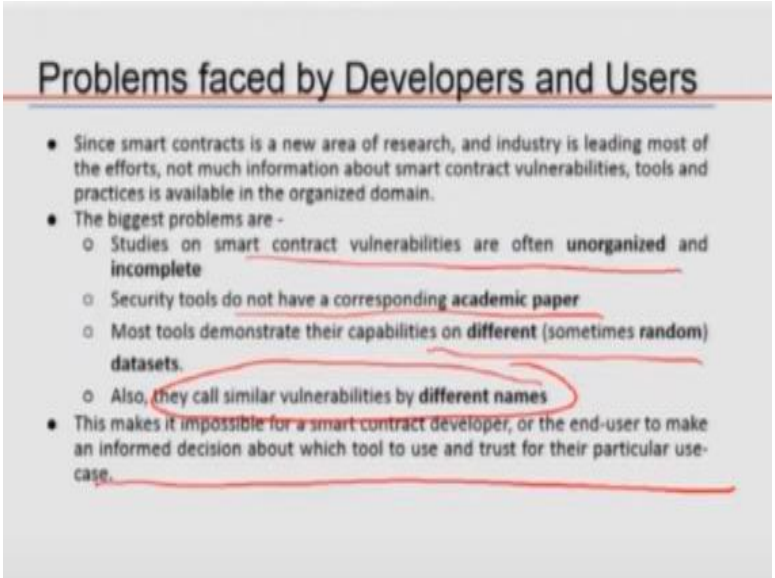
had ether balance in their multisig wallet could not get to their ether. And it was a big mess, right? So this was the Parity bug.

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)



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.

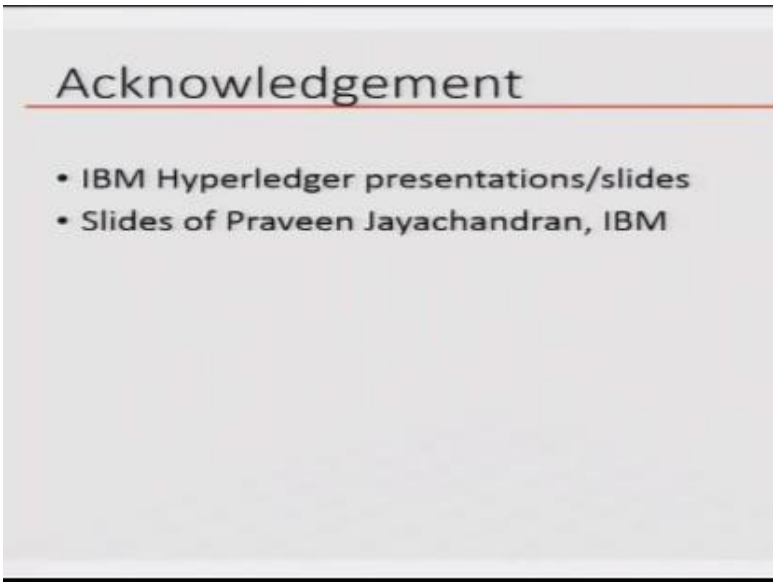
Introduction to Blockchain Technology & Applications
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture - 19

Welcome to another session of blockchain technology and applications on NPTEL. So far we have been talking about the permissionless blockchain, which are mostly cryptocurrency blockchains. So what we are going to start in this session is the first case of a permissionless blockchain and this is one of the most used permissionless blockchain in the industry right now.

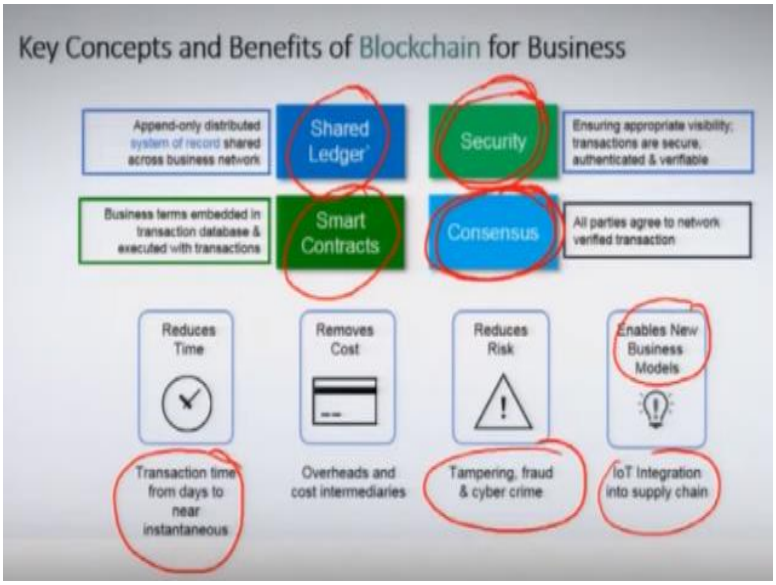
So this is the Hyperledger, which is actually propounded by the Linux Foundation. And in this session particularly we will actually go very superficially over the idea of Hyperledger what it is and what is the reason why it is so different from the bitcoin Ethereum type of blockchains. And then, in our next session, what we will do is that will actually go into more nitty gritty details of the Hperledger.

(Refer Slide Time: 01:18)



So a lot of the material in this session will be borrowed from IBM Hyperledger presentations and slides from Praveen Jayachandran from IBM.

(Refer Slide Time: 01:28)



So the idea is now we want to move into the blockchain for business. So what happens in a business and we already know that currently, when we have a system like a ecommerce system or grid management system, or payroll management system, usually they are centralized databases, and with a web server or some kind of a user interface, and then through web client or other kind of clients, we interact through the server to the database right.

So as we know that, that has an issue that the entire data is under the control of the company or the business that is hosting that server and the database server and therefore, there is lack of transparency. And also there are security issues, because of the centralization, there would be security issues, and certainly, there will be issues related to when you do not trust the particular entity that is hosting the application.

So that those are the reasons why a blockchain based solution which maintains the data integrity and also if designed properly and if the code is written properly, probably has quite a bit of better security is very good idea. But the other thing is that there are many such systems like client server systems, in which various parts of your business are hosted.

For example, you can have a university payroll system in a database server and the corresponding client server system through which it is interacted with. You might have the grid management system in another system. You might have the employee

HR related information on another system. And these things actually make the entire business process very fragmented and human dependent.

So what I want to do now is that I want to integrate all this into a single system, which is distributed, and because it is distributed, it has many players interacting with it. But earlier in a fragmented system where the people who are working on the payroll and people working on the HR database or people working on the student database, etc. they are different people.

They are full owner of those systems and any information that needs to go between these systems are usually done manually or through some kind of a offline mechanism and that basically leads to inefficiency. So what we are trying to now do is that we want to do a distributed information system and there are players or participants who will be all interacting with the same system.

And therefore, the notion of trust becomes a lot more important because we have many entities that are interacting with the same system. And we want to make sure that whatever it goes into the system is not tampered with by any of these entities. It is also want to make sure that they are not making any kind of changes or any kind of updates that are not acceptable to the other entities that are also using the same system for their information.

So all these things together will make this problem very challenging. So what we are doing is that learning from bitcoin and the Ethereum and smart contracts and all these ideas, we learned that we can create distributed system of records, which can be shared by many entities and these entities could be permissionless, which is the case with the cryptocurrency blockchain.

Or they could be actually permissioned, which means that everybody has some kind of a digital certificate or some other way of identifying themselves to the system, which makes it slightly better in terms of attributing who has made which transaction and then if they have done something wrong, then that can be traced back to that particular entity.

But having the ability to trace back or non-repudiation does not mean that they will not do bad activities or unwanted activities, because they might be doing it intentionally, or they might be infected by malware, and therefore very targeted malware, and they could do things like that. So therefore, I need some way of making sure that the entered data remains consistent across all the parties all the time.

And then we also want to make sure that any data that has been recorded remains untampered for once it is recorded for the future time. And therefore, ability to do that would come not by a database that is being connected to by a multiple of all these entities, but creating something like a blockchain, which is an append only distributed system, where integrity is very well designed.

Other thing is that many times across these different domains, different databases, people might have different kinds of business, process related data forwarding, data sharing, and many kind of functions that you want to call on a particular set of data, and then use the results for your computation and so on.

So therefore, normally if you are three disparate four disparate systems, then you have to somehow get the data from each of these systems and then try to do the computation on them and then put it on the other one. So in this case, you can automate all this through the use of smart contracts, right. So smart contracts actually can embed the business logic.

And that is another advantage that you are getting in a blockchain that has the ability to execute smart contracts. Now security is another issue and as we have seen in the previous sessions that security is not a given for blockchain. It is not like blockchain means it is secure. So you have to make sure that smart contracts are secure. You have to make sure their execution engines are secure.

You have to make sure your implementation is secure and all that things but that is something that you can probably do if you know exact what security properties you want. On the other hand, if you have three, four disparate systems, which are talking through manual intervention with each other, then the probability of making them secure is lot more difficult.

And finally, the consensus is something that even though you are we are distributed, and we even though we have parties that are involved in doing transactions or updates and fetching information from this distributed shared database, the parties will not do things unless everybody agrees or anybody who needs to agree, must agree to put something into the database or do make an update in the database.

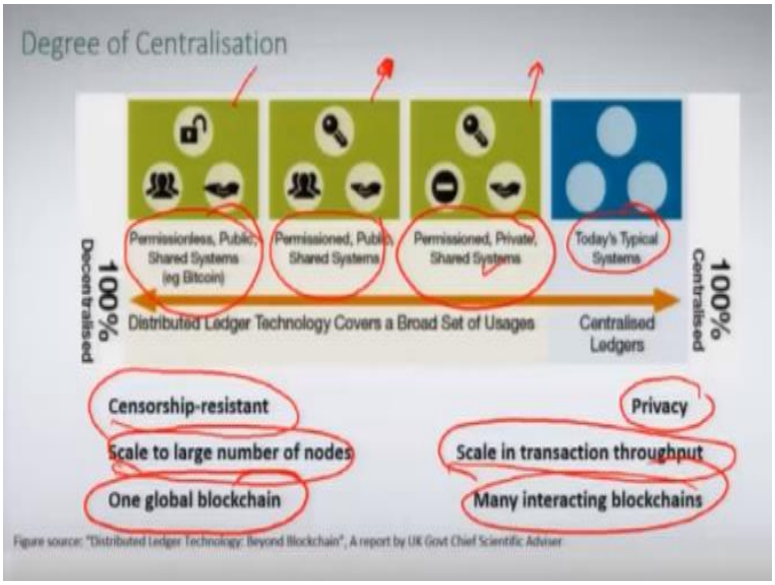
So consensus is important. So shared ledger, smart contracts, consensus, these are the things that we want to have and therefore we want to go for a blockchain based solution. Security is something that we claim to have better chance of making things secure is because otherwise, multiple disparate databases being mediated through other techniques, including manual techniques would certainly leak out data and so on so forth.

So the advantages that one could look for in such an implementation is that the total time should reduce because we are connecting multiple different data sources without manual intervention. I am automating the business logic through smart contracts. So I can assume that the things would be faster.

It removes the cost of intermediaries and various kinds of overheads when you want to have this interaction that is manual, or some cumbersome way of interaction. The tampering fraud and cybercrime can reduce if you design it correctly. Tampering is certainly property, tamper resistance is certainly a property of the blockchain. But fraud and cybercrime probably can only be reduced because of the transparency that you get.

So fraud is harder to carry out and then cybercrime also the same way. But again, I want to caution that these things are not just by virtue of being a blockchain, they do not come for free, they have to be properly designed into the system. And then new business models like IoT integration, etc., can also be made easier through this.

(Refer Slide Time: 10:33)



So as I said before, is that in today's typical system is centralized right. So you have database in which everything is logged, and every kind of updates, reading of information, creation of information is kept in a centralized system. Whereas in the blockchain spectrum, so if you go permissionless, public shared bitcoin shared systems like bitcoin Ethereum etc., then you have one kind of solution.

But then in a business scenario, the idea of having all the data on the blockchain to be public may not be very palatable to the industry people. On the other hand, you can also do permission, but public shared system. There also again, the privacy would be an issue, but it is permission. So attribution etc., would be easier. But if you have a permission, private shared system that is where this Hyperledger is placed, right.

So this is kind of bitcoin. This could be a private instance of a Ethereum, where you can instantiate a private Ethereum network for yourself or for your organization. That would be this and this would be Hyperledger. So some of the things about ledger technology as articulated in this document, Distributed Ledger Technology: beyond blockchain. It is a report by the UK Government chief scientific adviser.

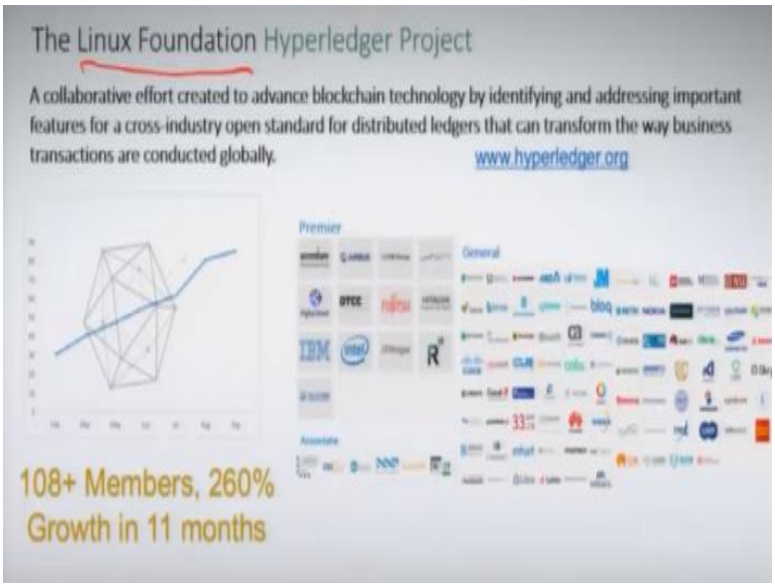
And there they basically say that the blockchain technology is a transformational technology for the upcoming future and there are certain things that you can get out of blockchain that you cannot get in the current systems. So censorship resistance is one such property. If you have something in the blockchain, it is replicated everywhere. So you cannot really you know suppress that information easily.

It scales to large number of participants. So that is one thing. You can also have one global blockchain or you can have many interacting blockchains. You can have privacy if you are going for this kind of a system. And the transaction throughput can also scale if you have a permissioned private blockchain compared to let us say bitcoin, where we saw that transaction throughput is very low.

You have about one block per 10 minutes, which is not really fast enough to do global scale business. So all these things are something that you can get from blockchain. And in particular in the business scenario to in order to integrate various information systems, which are normally run as separate client server system in a blockchain based solution, they will all come together and then because it is permissioned and because it is private privacy and other things can be done in a much better way.

And since it is permissioned and private, you do not have millions of nodes and therefore, you are going to get better scalability and throughput. And also you can, if multiple business entities are interacting through the blockchain, then one entity cannot suppress the information of another entity unduly without everybody's consensus.

(Refer Slide Time: 14:00)

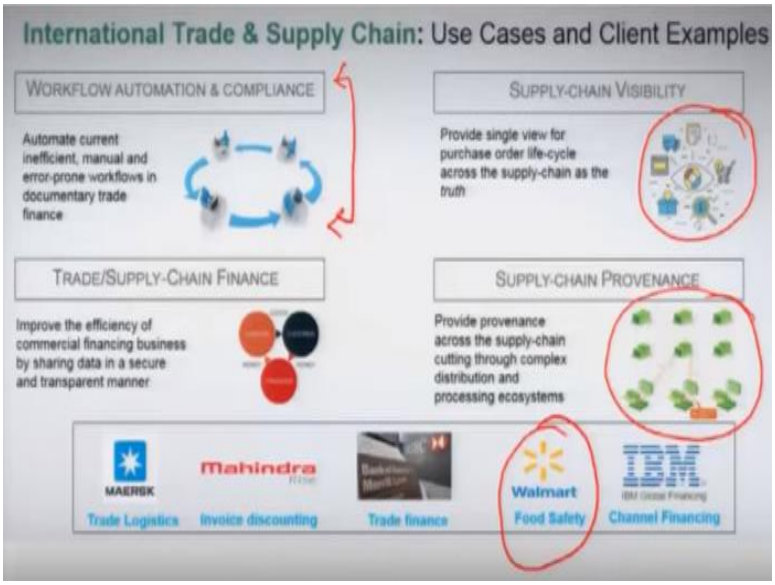


So with these in mind, Hyperledger project came into being. It is actually some champion by Linux Foundation. And they have multiple different projects, right? So they have Hyperledger composer, they have Hyperledger fabric, all kinds of other

family of products that are available. But they are based on the basic technology underlying Hyperledger is the same, whether it is composer or whether it is through fabric.

But the idea is that how easy it is to set up a blockchain and how easy it is to actually write create an application on top of that blockchain varies between, let us say the composer versus raw fabric.

(Refer Slide Time: 14:44)



So some of the used cases that IBM has already done is, as you can see that none of this has anything to do with cryptocurrency. It is all about creating the business process in a seamless kind of IT system using the blockchain right. So workflow automation and compliance is something that has been tried and then the idea is that when you have a workflow in terms of a business process.

For example you have a workflow that customer comes and makes a negotiation, signs a contract, then certain deliveries are done, deliveries are tested, testing is then reported for compliance, then the payment is done. All these things usually is done from desk to desk or from office to office in separate systems. Now in a blockchain all these things can be automated into a single integrated system.

And therefore, things can be done in a much quicker fashion. Supply Chain Finance. For example, you know you have suppliers, you have customers and you want to have

a financier involved for example, and let us say a bank through which the customer takes a loan and then it gives money to supplier.

Then customer pays monthly payment and all this stuff has to be together because the supplier if the customer fails to provide let us say the down payment or customer fails to make proper payment then supplier might intervene and make sure that the service gets stopped or something. So all these things you know is usually done by you know multiple different systems the one that is between supplier and bank.

One that is between supplier and financier and one that is between supplier and customer usually are separate systems. But now with the blockchain you can all put them all together in the same system. There will be everybody will know what is going on what kind of transactions are going on and then the supplier can easily make decisions based on the customer's compliance with its obligations.

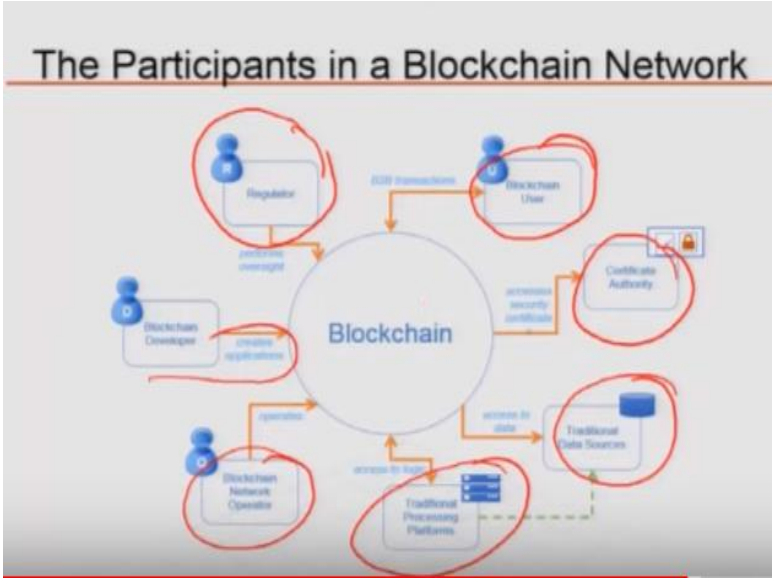
Financers similarly can know whether the supplier is providing the goods to the customer and so on so forth. Supply Chain Visibility and Supply Chain Provenance, right. So supply chain is often like for example, in Walmart, the let us say the food that comes from many different sources right? So meat get packaged by various companies at various plants.

And they get supplied to Walmart in their grocery section. Now the question is that suppose some particular batch of meat turns out to be bad imported by multiple customers, then one has to know where that meat came from and what are the other Walmart stores which has the same batch of meat and then they have to be taken off the shelf and corresponding payment and reimbursement and all that stuff has to be done by the supplier.

So usually, this is also tracked very manually in normal scenario. But with a blockchain implementation all that stuff will be in the same blockchain and then all the necessary visibility as to where something is coming from how to attribute it to particular supplier. And whether the supplier is giving credit to the store for returning the meat or throwing out the meat because it was the fault of the supplier.

All that information can be actually gathered from the blockchain. And it cannot be tampered with by any of these parties. And that is a good thing to have. Similarly, the supply chain provenance, like if you want to know where something came from, whether it involves child labor or not, all that stuff can be all tracked if every step of the process puts in all the information on a single blockchain. So these are some of the applications we are talking about.

(Refer Slide Time: 18:52)



So who are the participants in such a blockchain network, right? So in case of a bitcoin we saw that everybody is similar, right? So some of them distinguished themselves as miners because they have more computation power, but everybody has the almost the same ability to do and things on the blockchain right, accepting that some of them may become miners, same thing with Ethereum.

But we also saw yesterday that because of the inequality that has happened over time, so therefore, it is not really the same, but at least as far as the technology is concerned they are at the same role and responsibility. Here actually, it is quite different. So we have the blockchain. So blockchain developer creates applications. This is also true in Ethereum that you can be a smart contract writer.

So you are a blockchain developer. And then, blockchain network operator is the one who maintains the blockchain network, which means that all the entities that maintain the blockchain, do the consensus everything that has to be maintained by some entity. So that is a network operator. You have the blockchain user. For example, if it is a

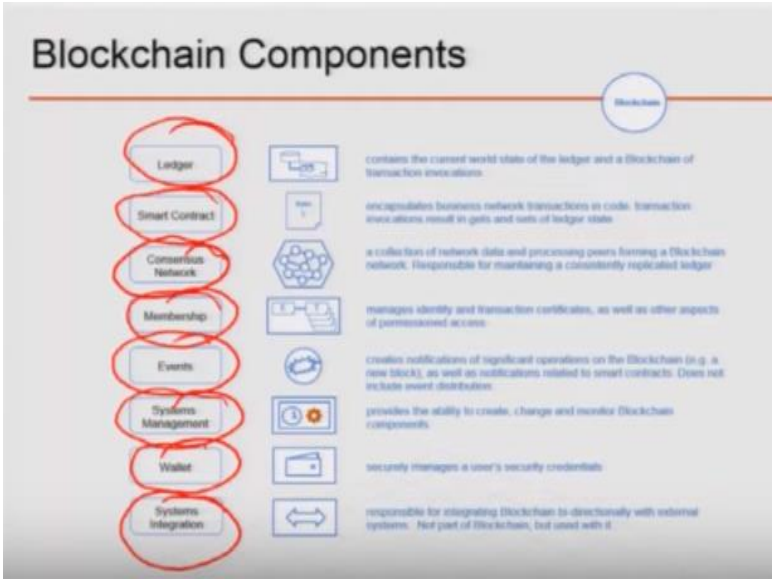
supply chain blockchain, then the blockchain user would be all the entities that are involved in the supply chain, the customer at various steps and so on.

Now in a permission blockchain, you also need to identify users with some digital certificate. So there has to be a Certification Authority involved. And this could be an internal Certification Authority or it could be a regular, you know digital certificate provider CAA, that kind of entity. Now also blockchain might be used to interact with the legacy traditional data source.

So that will be actually the blockchain might actually pull data from the traditional data source to have the information that is required to actually do execute some transaction or it can also send information back to traditional data source and then the traditional processing platforms that was part of this traditional data source based IT system that the company already had, could also have this thing integrated with the blockchain.

And then the regulator, so in case of this, this is the supply chain, for example, or some kind of a money lending application over the blockchain. Then a regulator can also look at all the transactions he has been, the regulator will be permissioned into looking into the blockchain. And then therefore, they can check the regulatory compliance much more easily than today when there is huge amount of paperwork involved. So that is the vision of this kind of Hperledger based blockchain applications.

(Refer Slide Time: 21:57)



So what are the components? Components is, obviously there has to be a shared ledger, which contains the current state of the ledger and the blockchain of the transaction invocation. So the state in this case is not really on the blockchain, but it is in a store that everybody has a copy of. So everybody knows the state of the blockchain.

But the blockchain itself has enough information in a hashed form that will make sure that the integrity of that state information is not compromised. And because of the replication, it cannot be easily compromised anyway. Smart contracts are the actual programs that will run to make transactions happen. And they actually have the business logic, right.

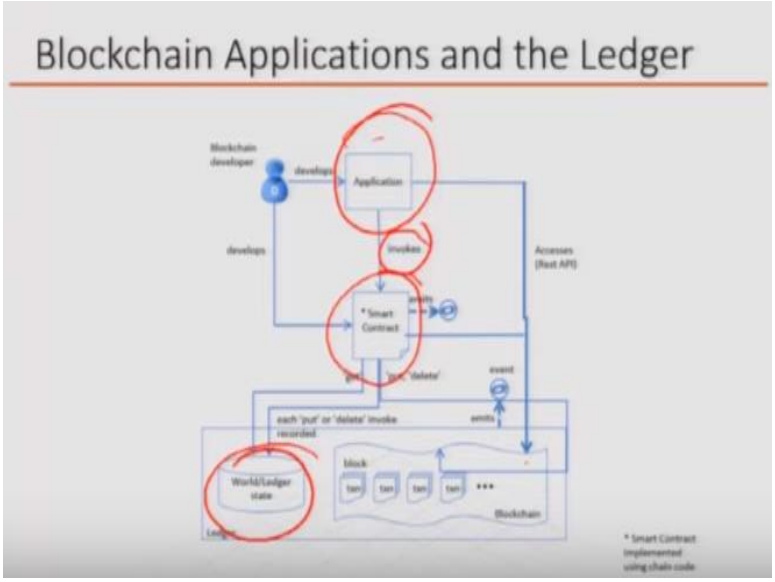
So by invoking multiple different transactions in the smart contracts, a particular business process, maybe automated. Consensus network, so since it is a blockchain, there has to be a consensus. And then the consensus network basically is responsible for that membership. Of course, this is a membership service is basically certification or identification for each of the participants that has to be there.

Events. So blockchain transactions may actually create certain events and those events might be also used for triggering other transactions. So there is a notion of events and then the system management because there is a network operator who can create change monitor various blockchain components. User wallets in this case, since we are not talking about cryptocurrency blockchain, it is not about money in the wallet.

But wallet actually contains the user's security credentials, which could be digital certificate, it could be username, password, or whatever the information is. And then systems integration is this basically the integration with the external systems, the systems that are legacy systems, systems that are not participating directly in the blockchain. So in a blockchain like Hyperledger, these are the components.

And with the very well defined and different roles and responsibilities, which is quite different from the blockchains that we have seen so far.

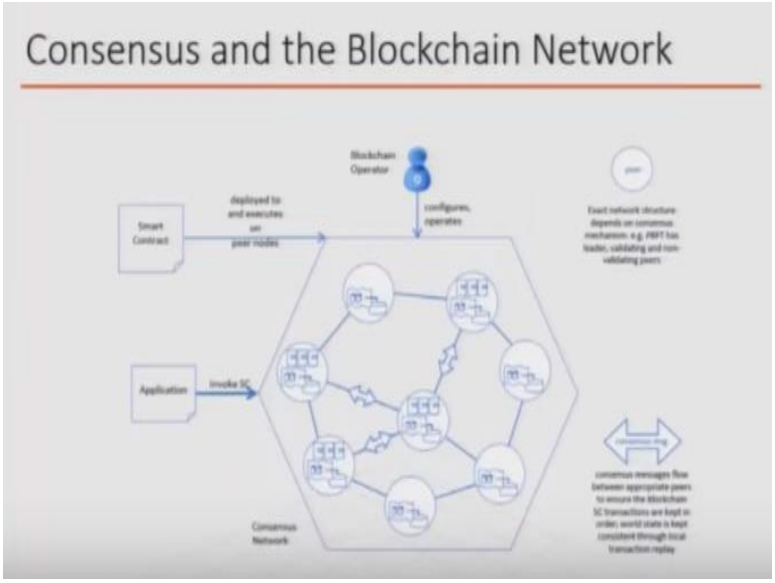
(Refer Slide Time: 24:18)



So the blockchain applications actually invokes smart contracts. So an application as we saw is like dapps, right, a distributed application, it has ability to also query the blockchain. And this access has to be provided through what we call rest API's. And then, the blockchain developer develops both the application and the individual smart contracts that together make the application work.

And then there is a state of the ledger. And that is a separate database, but everybody has a replica of the blockchain and this state information at all the different nodes. And that is how the entire system is configured.

(Refer Slide Time: 25:11)



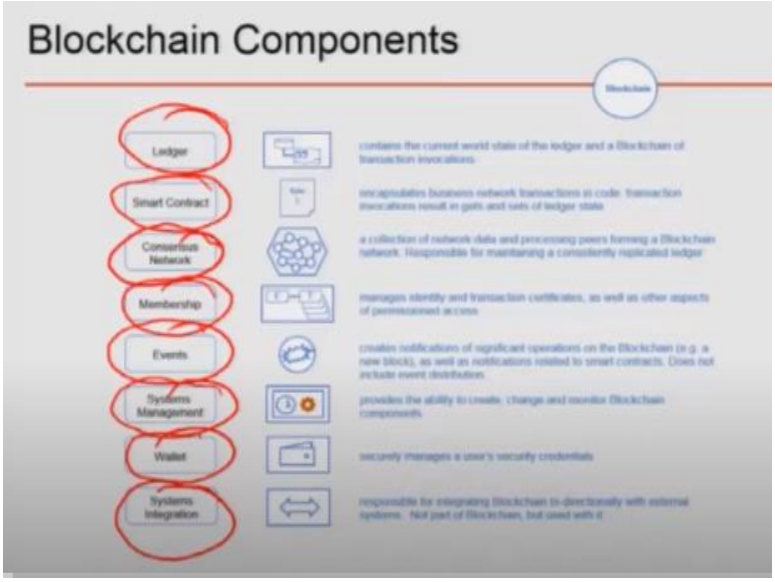
So when we come back, we will talk more about the functioning of this blockchain. So far we are only talking about who are the participants who writes the application, who writes the smart contracts, what is in the blockchain versus what is in the state ledger, state part of the ledger. And then we talk about roles and responsibilities. But when we come back, we will talk about little bit more on how the entire Hyperledger blockchain normally functions.

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

Lecture - 20

Welcome to another session of blockchain technology and applications.

(Refer Slide Time: 00:19)



So last time where we stopped was a very superficial overview of the Hyperledger fabric, which is a permission blockchain, a private blockchain and we were discussing what are the different components. So we talked about ledger, where the transaction history is hash chained for integrity checking. We talked about smart contracts and smart contracts in case of Hyperledger is, is can be written in any language.

And currently, most of the smart contracts for Hyperledger is written in Go language, but the way Hyperledger executes the smart contracts is inside a Docker and they do not use a special EVM for this like Ethereum uses EVM and therefore, it is for a specific language for which bytecode has to be generated. In case of Hyperledger, any general purpose language can be used. But most common is the Go language.

And then of course, there should be a consensus framework. And as we will see that in Hyperledger, the consensus is pluggable in the sense that you can decide what kind of consensus algorithm you need and accordingly you can plug that in into the

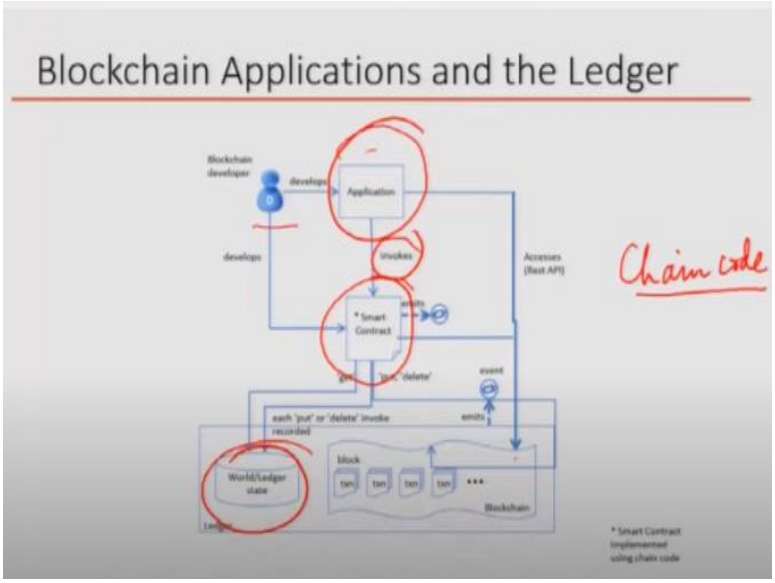
Hyperledger framework. There is also something called membership because we are talking about permission blockchain.

So there has to be a way to prove your own identity. And therefore, there is a membership which is pluggable. So you can have a digital certificate authority connected or you can have your self-signed digital certificate system or you can have a default membership system that obviously is not very secure, this kind of stuff. It also has the notion of events.

So you know like any other smart contract, the smart contracts can actually generate events, which may actually trigger other activities including invocation of other transactions or smart contract functions. Then of course, the system has to be managed. So there are chain codes or there are contracts, smart contract like programs that does some system management functions.

So that is the system management code and then wallet in case of Hyperledger is about this, you know the account and the corresponding identity that is given to it by the membership service and then the entire thing has to be integrated, because, you know you can customize your particular implementation of Hyperledger with a different permission membership service with a different consensus service and therefore, you have to do an integration.

(Refer Slide Time: 03:29)

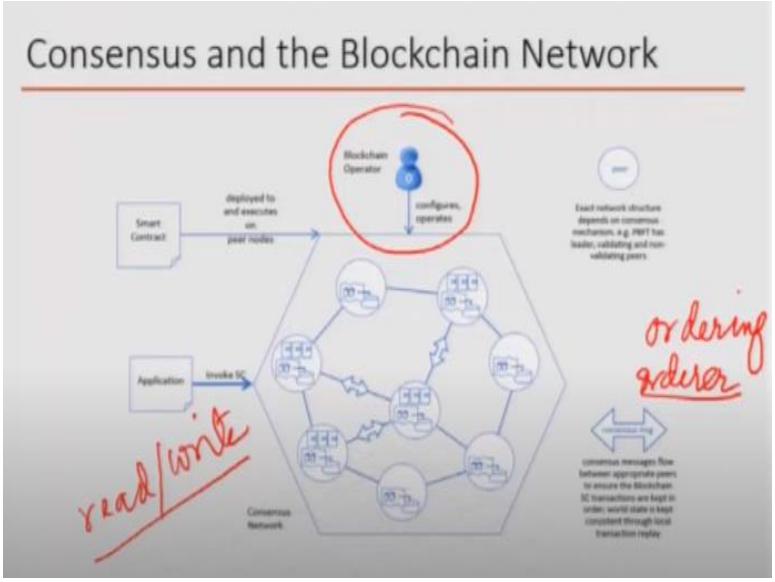


So what happens in this is that, a blockchain developer develops an application which invokes smart contracts. In case of Hyperledger, they are called chain code. So since this is blockchain so the code that runs in the blockchain system is they call it chain code. And then there is the blockchain and the state of the blockchain which is kept in a database.

It will be more clear as we go deeper into these Hyperledger architecture and system description, but the idea here is that the developer develops an application which is capable of doing transaction, but before doing transaction it has to verify or prove its identity and then it can invoke through the transaction request it can invoke smart contract which are chain code.

And then that once that is done, there is you know the transaction has to be persisted for integrative verification in the future for audit function and so on. And then the state of the system has to be also stored because next time you invoke a transaction, it has to be from the right state.

(Refer Slide Time: 04:50)



So here is the consensus mechanism. So unlike the Ethereum or bitcoin where the consensus happens first through proof of work or proof of stake or whatever consensus mechanism is inbuilt into the system. And then the consensus what it does is that everybody who is interested in building the consensus, they are called miners.

And what the miners do in such systems in case of bitcoin and Ethereum etc., is that they select the transactions that they want to put into the block, they execute them and make sure that the transactions are valid by looking at the signature etc. And then once they execute them, they actually try to solve the whatever proof of work or proof of stake problem.

And then whoever wins among all this miners, whoever wins will be doing the next block. so the next block will be percolated everywhere. And then every node will accept or try to diverge, but eventually everything will come to a same main chain. In Hyperledger, however, that is not how it is done. So there are certain nodes called orderer nodes. So ordering nodes or orderer nodes, which are actually responsible for the consensus.

The other difference is that the consensus mechanism is all about the ordering the transaction because when you do multiple transactions from different clients, if you do not order them correctly, then if the different nodes order them in their own way, then what would happen is that the result of the set of transactions will be different at different nodes.

So the way this is solved in bitcoin or Ethereum is that eventually only one block wins. So the order that is selected in that particular block is what is you know percolated and persisted everywhere. In this case, this is not how it is done. So what happens is that you order the transactions using this ordering service and this transaction ordering is actually done by solving a consensus.

There are only certain nodes which are involved in that process. And once they do the ordering, they will now create the block, create the hash and then send it to all other nodes. All the other nodes will check whether the transactions that are in the block that they are receiving are valid. So note that here the person who is doing the consensus or ordering is not even checking whether the transactions are valid.

His or her only job is to check is to order the transactions and then agree. If there is multiple nodes which are involved in this ordering process, they have to all agree or have a consensus that this is the order that we are going to persist. And then they will

send it to everybody else in a block and the others are responsible for validating that the transactions are valid.

If they are not valid, then we will mark those transactions which are not valid and persist the rest and then they will change the state of the state store according to what those transactions would achieve. So each transactions will change the values of certain states, right. So those values will be changed in everybody's database.

So you see a big difference here, that you are giving the consensus activity, you are giving the responsibility of consensus to a different specific set of nodes, which we call the orderers. And they all together form what is called the ordering service. And these orders are not checking the validity of the transaction. They are only deciding the order of the transactions.

And then they are telling everybody that this is what the order should be. And this is the block of transactions. So you should all check that the transactions are valid, and then you will persist it. And then all the other nodes will do that, add that node, add that block to their blockchain copy, and then change the state and that is how this thing works. So here you see that the application invokes a smart contract.

Every node, many nodes will be executing different set of smart contracts depending on what they are doing. The smart contracts are actually executed inside a Docker. So each smart contract runs inside its own Docker and this Docker makes it possible that the smart contract can be written in any programming language. So it does not have to be Go.

It can be C, C++, Java, whatever language you prefer, you can write the smart contract. Smart contract is basically the application logic. So once this application invokes this smart contracts, then the smart contracts will execute. And then it will realize that if I execute this transaction, then this will be the change in the state. These variables, these state variables will change.

For example, if they are keeping track of money, then how much money is being deducted by a transaction, which account gets that money, all that information, they

will gather. But they will not persist it because persistence time is later. So they will send back to the application the result of executing those transactions. So in that sense, we say that they simulates the transaction. They really do not execute the transaction.

So they simulate the transaction. They know if the transaction happened, what would be the result, they will send the result as what we call a read/write set. So what value is read from the state and what value will be written. For example, if they are keeping track of money, then which account has to be whose balance has to be read, and whose balance has to be written to.

Maybe both the balances have to be written to because you are say you are sending some money from this account to that account. So both accounts have to be read. And both accounts have to be written to. So that read/write set will be sent back to the application. And then the application will then send this information to the ordering service.

The ordering service will now will be getting similar transactions and their read/write sets from multiple different clients and therefore, they will put them together in a block. But before they put them in a block, they will decide what is the right order. And to decide the right order, they will go through the consensus mechanism. And once they have decided the right order, they will create the block, they will compute the block hash and then send it to everybody else.

There is a sequence number that they will also provide to the block. But that is more detailed implementation to avoid confusion, if the same block is sent multiple times etc. So confusion avoidance is done by this sequence number. In any case, the order will then send it to all the others or what we call peers.

And these peers will then check the validity of the transactions and then they will go ahead and only valid transactions, they will actually persist and then they will check the read/write set and then they will change the store. So they will change the blockchain by adding a new block that was given to them by the orderers and then

they will change their state, database to reflecting the read/write sets that came with these transactions.

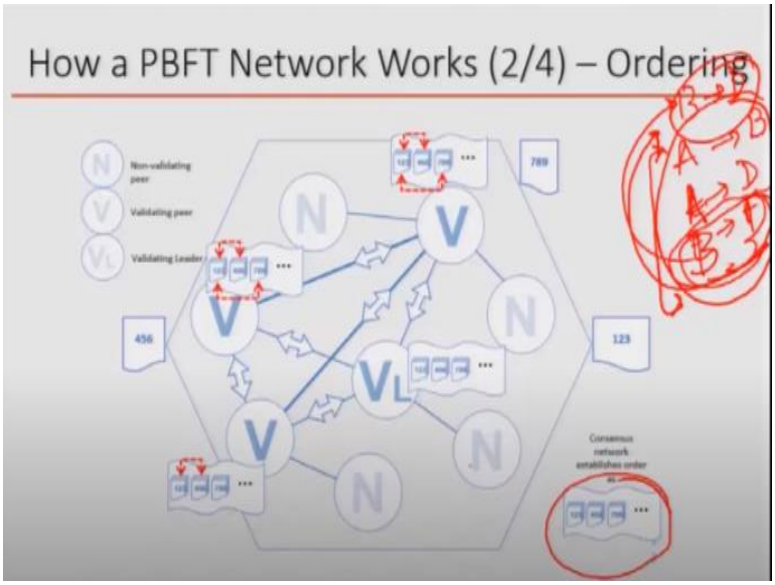
So this is how this works. So blockchain operator actually creates. Why is there a blockchain operator? Because it is a private permissioned blockchain. So let us say I am State Bank of India and I am keeping all my transaction reports in this blockchain. So therefore, The State Bank of India's IT department or somebody will set up the blockchain.

They will set up the chain codes, they will set up the applications. And then the clients, that means, you and I, when we actually do a internet transaction will be actually when I fill in that I am going to transfer this amount of money to this other person's account, that information will invoke a suitable smart contract. And that smart contract will give me back the information about what will happen if this transaction executes.

And that information in a read/write set will be given back to me. Back to me means in the background, I will not see it. It will be the code that is running on behalf of the State Bank at the backend of my let us say web based front, web based client side application. Then the back end that is behind my client is going to now collect these and send it to the ordering service. The ordering service will order them.

Then send it back to their all the other nodes. These nodes are there to create replica of the blockchain. And the replica is needed, they could see currently, you know everything is probably put in into a database, which is a single database. Maybe it has a single replica because of the backup and so on. But now there will be many replicas, many peers, many machines on which this whole thing will be persisted. So that is the idea.

(Refer Slide Time: 15:52)



So if you look at this, so this is basically the ordering. So let us say one client submits the transaction with a transaction ID of 456. There is another client here at the same time he is doing also another banking transaction and his transaction ID is 789. And another client which is also probably checking balance or doing something else some transaction, that will be let us say 123.

So each of these will now send this information to this ordering service nodes. So these are the ordering service nodes that they have sent to. Now ordering service nodes they will this means here is the ordering messages that the ordering nodes or consensus nodes will exchange with each other in order to achieve a consensus. And at the end, they will achieve a consensus and then that information will be sent to everybody else.

So look here, that there are nodes which are not part of the consensus. They are just there to keep the replica of the blockchain in order to have enough redundancy in the system. So they are not doing the consensus. And that is a major difference with the regular block chains that we have seen earlier that everybody can be part of the consensus mechanism. But here certain nodes are configured to be part of the consensus.

Some nodes are configured not to be part of the consensus, but keeper of the blockchain. So now these things will be ordering these transactions. So the lot of messages will pass and then a consensus will be reached about the actual ordering.

And let us see, eventually they decide that this is the order 123 first although remember, like when we were describing, we said that 123 came slightly later, etc.

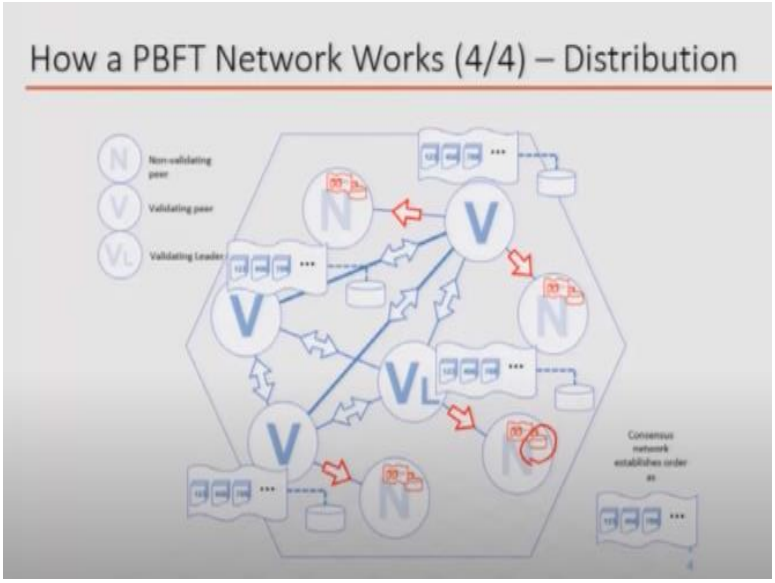
But we are doing this in the correct ordering. Now if 456 let us say this transaction is about two clients A giving money to B and this one is C giving money to D and this one is E giving money to F. Then any order would have been fine because there is no conflict between them. But suppose this one is A giving money to B.

And this one is A giving money to D and this one is B giving money to D. In that case, one has to decide an order. For example, A giving money to B would mean B's balance will go up. If this was put before and let us say B did not have enough balance earlier, then this transaction will not be valid. But if I order it this way this transaction can be done. So therefore the ordering will matter if there are such conflicts.

Otherwise the ordering could be any ordering, but the consensus network will decide what the order should be. And if the order they choose still remains the same, then maybe when these guys get the blocks and they will say oh B to D, let us say came first, let us say this is what the orderers decide. Then they will say this transaction is invalid because B cannot give money to D.

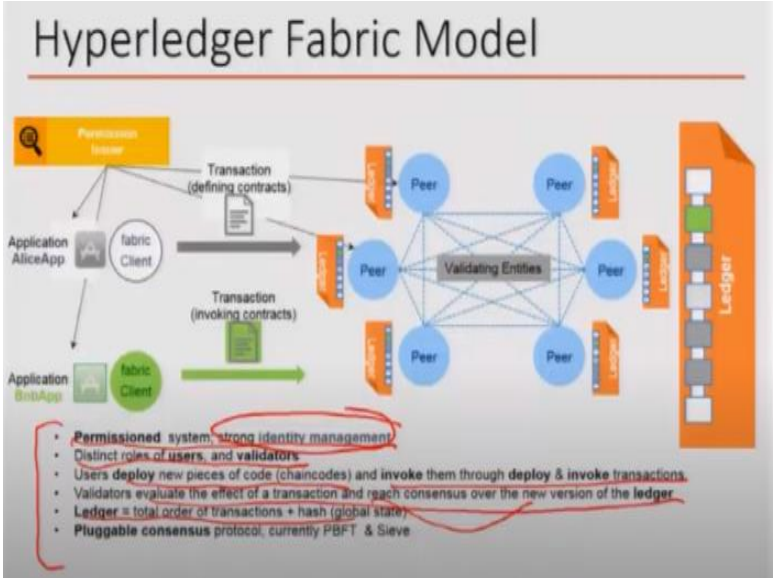
So then this client has to again put the transaction back after he gets the money from B. So the client or the customer will may actually get a denied transaction and he tries after some time and he will get a fulfilled transaction, because in the first time he tried at that time the ordering service ordered it in a certain way. This can happen.

(Refer Slide Time: 19:51)



So then once this ordering has been done for this set of transactions, they will be sent to all the peers. And all the peers will then add the new block to the blockchain and also change the state of the state variables in the databases.

(Refer Slide Time: 20:10)



So that is the idea. So the basic point here is that Hyperledger is a permission system. So there is a strong identity management, everybody has an identity that they have to get either from some kind of a Digital Certification Authority. Or if they are all part of the same company or something then the company can provide the self-signed certificates or any other mechanism by which a person's real identity will be connected to his identity here.

Again, you see in the permissionless, I can create a pair of public and private keys and I can say this public key hash is my identity. There is no connection to that of my real world identity. Here there will be a strong identity management. So your real world identity would be connected to your identity in the system.

For another difference with these previous ones is that in case of Ethereum, or in case of bitcoin, what we saw is that every member or every participating node or every participating member of the blockchain have the same role. Now some of them might actually say I do not have enough computation power, I will not do mining. And they just do a light, they just maintain a lightweight client, but that is their choice.

So but they could if they have the right if they can try to mine, and they might not ever win because they have only just a laptop, but they can do everything that anybody else can do, except that they may not have the right computation power. Here, the blockchain is configured such that all the users have different roles. Some are just clients. Some are just peers, some are actually orderers or consensus making notes.

So everybody has their certain roles and this is pre-configured by this blockchain administrator. The users can deploy smart contracts or chain codes and invoke them through deploy and invoke transactions. Validators evaluate the effect of transaction and reach. Well, the validators do not reach the consensus. This is slightly misrepresented in this slide.

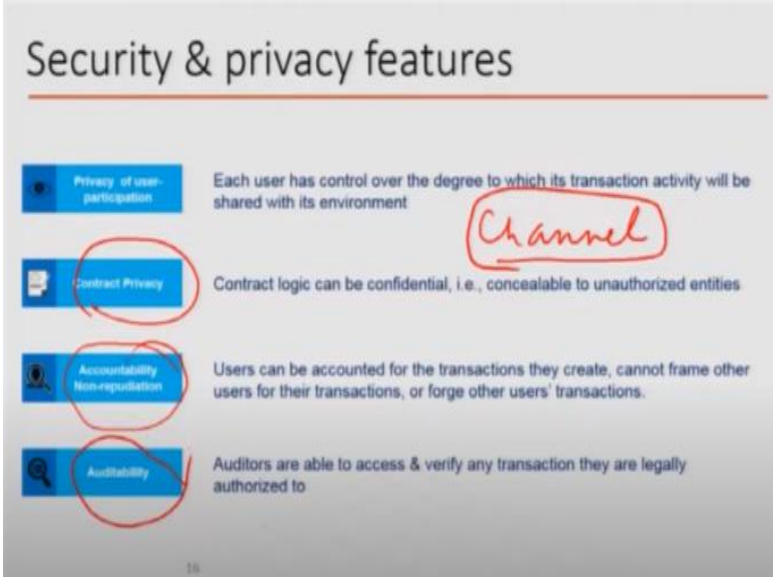
The orderers do the consensus and then orderer decided what should go as the next block. And then validators actually do the main job of validating the transaction and persisting the transaction. Ledger will eventually give you a total order of the blocks. So the transactions are ordered inside the block. But that ordering is what is decided by the consensus making nodes. And then the blocks will never diverge right.

So here there is no scope for creating forks right. So even in Ethereum or bitcoin forks can happen, but those forks will eventually you know converge, because after a while the longest surviving chain will be what others will add to. Here there is no such

scope, because there is every replica will be a single linear, you know chain of blocks. The consensus is pluggable.

And then there are multiple different consensus. We will talk about that as we discuss more details.

(Refer Slide Time: 23:38)



Then the question is that since every user's real world identity is connected to the actual identity. So what you might ask is that what about privacy because in blockchain, even though in bitcoin or Ethereum, even though all transactions are in the open ledger, public ledger, at least I can nobody knows who I am. So whoever I am giving money and etc., will be difficult to find out, although it is a pseudo anonymity not full anonymity, but what about in this case?

So in this case, normally, this blockchain is used for businesses right. So therefore, the privacy is not the same as privacy of individuals or in money transaction in bitcoin and blockchain. But even then, a group of users may belong to a particular organization and another group of users may belong to another particular organization and they are doing transactions on the Hyperledger.

Then you may not want that this what this group are transacting and doing should be known to the other group and vice versa, right. So therefore, this Hyperledger has a notion of channels. So channels basically means that in the same set of computing nodes, I can run a Hyperledger, but I can define channels. So certain users will only

transact on a particular channel, other set of users will transact on a different set of channel.

These two channels cannot see each other's transaction and they maintain a different blockchain. So let us say State Bank of India and UBI decide that all their interbank transaction etc., will be done through blockchain. So that there is transparency, auditors can see and everything. So what they can do is that they can have a channel in which certain nodes from both organizations are there.

Through which the only the interbank transaction type of transactions will be put in a separate interbank transaction blockchain. On the other hand for their internal customers, their own customers, let us say their transactions are being recorded. So there will be a channel that is SBI will have another channel that UBI will have and they will not see each other.

Only the third channel in which the interbank transactions are being put that will be kept. Now why is this channel why not two, three different Hyperledgers? Because they can share resources. For example, ordering service can be shared by all these three channels. The ordering will never mingle the transactions or blocks of each channel.

It will know which peers belong to which channel and then accordingly send the blocks that it cuts to those peers. So there will be separation and therefore, in that sense there will be institutional privacy. But individual privacy is not a big issue here because everybody is there working on a business and official capacity.

The other thing that the Hyperledger has compared to Ethereum for example, so in Ethereum, all these smart contracts are on every node, is actually on the blockchain. And every node that has a copy of the blockchain has all the smart contracts. And they also execute all the smart contracts, because that is how the ledger works.

They have to execute and make sure that the transactions are actually valid, and also they have to change the state. What we will discuss shortly is that in Hyperledger, this

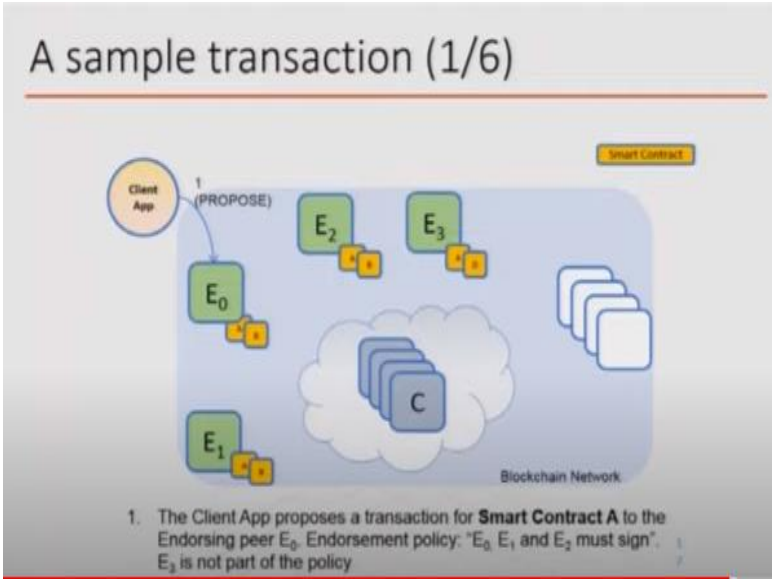
is basically means that my business logic and your business logic will be known to each other. In a business setup that may not be the good things.

For example, if SBI is having particular logic for deciding who to give loans for example, and UBI has a different set of logic and they do not want their business secrets to mingle. So in that case, this person's smart contract and that person's smart contract may not be, you know visible to each other. It is not desirable to them. So that can be done in this setup in Hyperledger setup, but not through channel.

You can be on the same channel, but you can have separate trust domains, and therefore, only people in your trust domain, the peer in your trust domain can only see your smart contract, others cannot see. So that contract privacy is another thing that Hyperledger gives. The accountability and non-repudiation comes from strong identity, right. So everybody has a strong identity.

So they if they put in a transaction, they cannot say I did not do this transaction. And therefore, there is also auditability. Auditors can access and verify transactions that they are legally authorized.

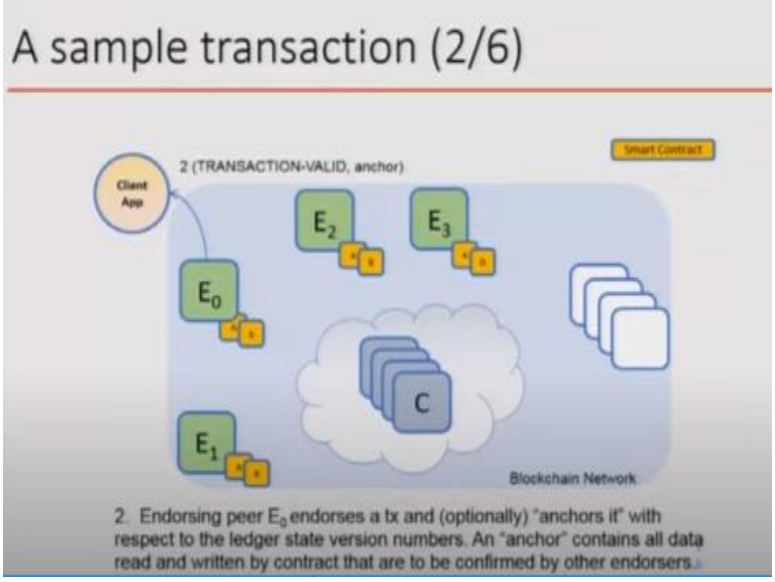
(Refer Slide Time: 28:56)



So a sample transaction would be like a client application. Like I am a customer to SBI and I am doing a transaction. So I log in through my client application and I propose a transaction. I say give this much money to this person. So then it goes to

certain nodes or what we call endorsers. So for each transaction when it is proposed, it has to be endorsed by a set of peers.

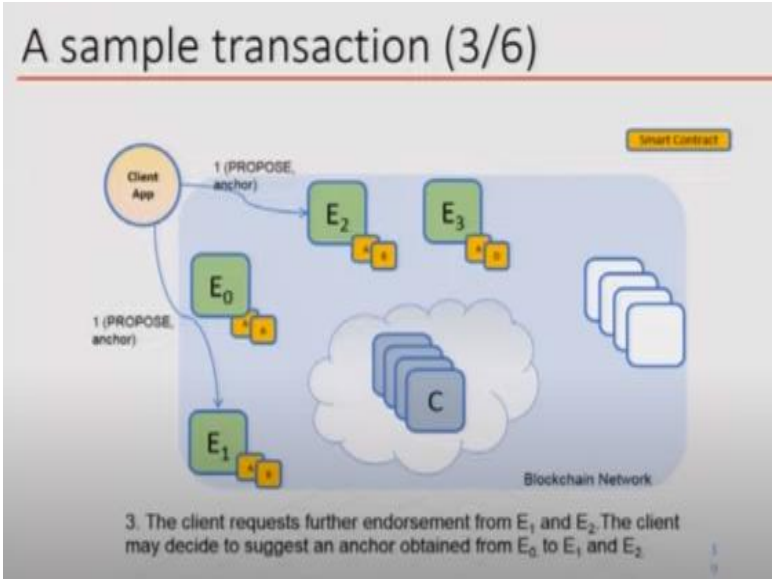
(Refer Slide Time: 29:24)



So let us say E0 is my endorser, and I only need one endorser. Let us say this is the endorser. It basically contains chain code, let us say the smart contract A. It also runs other smart contracts because it is a computer. It can run multiple smart contracts but for let us say, for transaction processing A is the smart contract that is required. So what it will do is, it will execute, simulate the execution of the transaction see if you have balance.

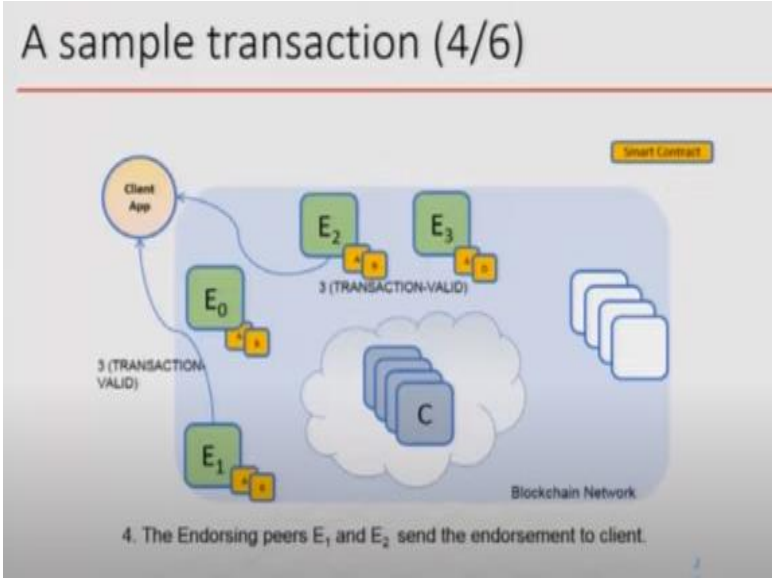
And what would be your final balance if you give that money, and what would be the final balance of the other account and it will create a read/write set and send to you.

(Refer Slide Time: 30:02)



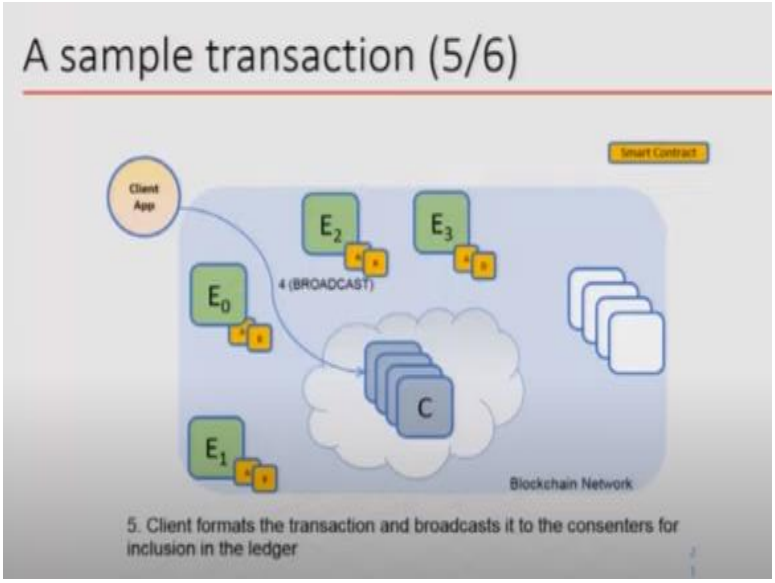
Then let us say the endorsement policy says that you have to get at least maybe two endorsers to say that what you are doing is allowed. So then it will send to the other nodes which are also in its endorsement policy.

(Refer Slide Time: 30:16)



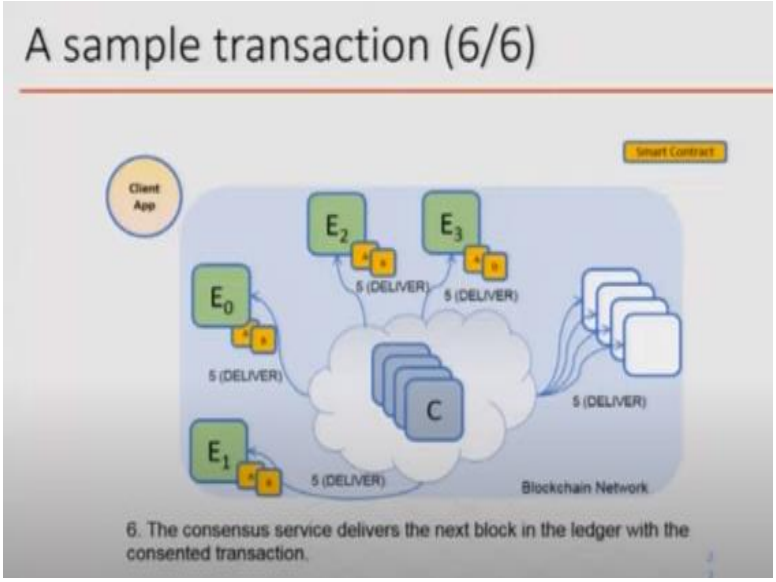
And then they will also send you that same kind of information that if you did the transaction, what would be the read/write set what would be the changes in the state.

(Refer Slide Time: 30:27)



So then it will send it to the blockchain's ordering service.

(Refer Slide Time: 30:33)



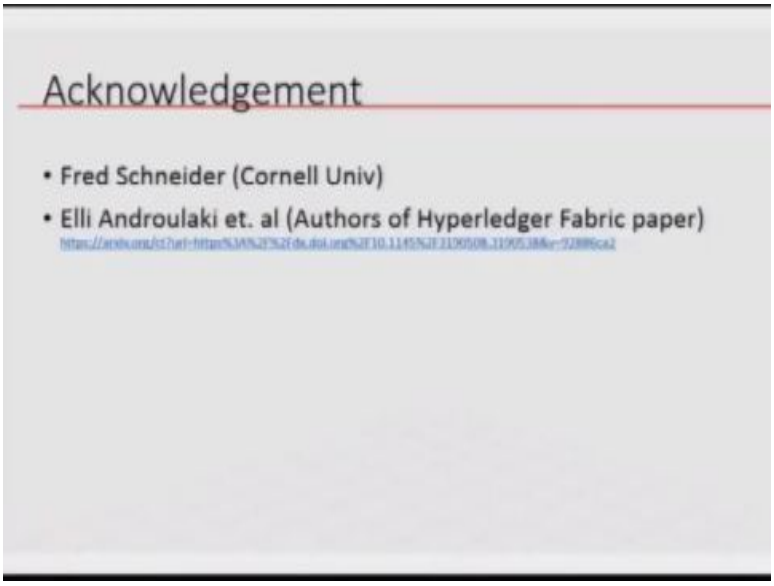
The ordering service will then create the ordering, and then it will deliver the blocks to all the other nodes and other nodes will then including these nodes who are the endorsers, and then they will all persist that blockchain. So this is an overall view of the Hyperledger blockchain. Now when we come back, we will go into a lot more details about how Hyperledger works and the more theoretical underpinning of the design of the Hyperledger blockchain.

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

Lecture - 21

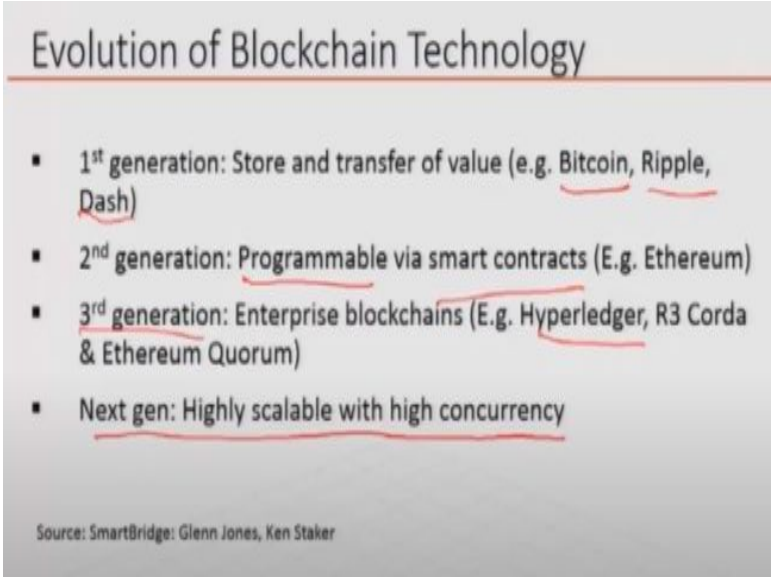
Welcome to another session of blockchain technology and applications. So last time, we talked about the superficial overview of how Hyperledger blockchain works. What we want to do in this session and the next session is to actually take you a little bit inside the theory that is behind the particular design choice that the Hyperledger people made. And if you understand these design choices, you will understand the goal of Hyperledger blockchain or similar enterprise blockchains.

(Refer Slide Time: 00:50)



So much of this material is from Fred Schneider and the original Hyperledger Fabric paper that is available in archive.

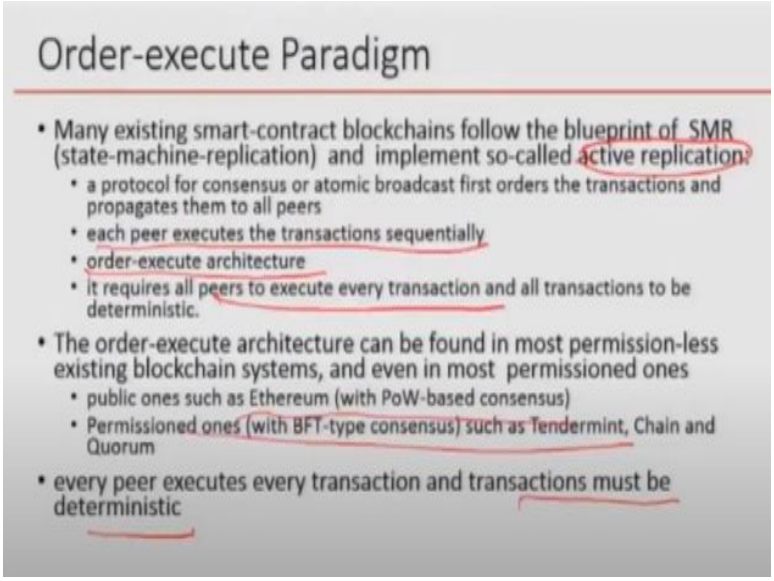
(Refer Slide Time: 00:59)



So we have been discussing this several times, but there is no harm in repeating. That the first generation of blockchain which basically means bitcoin and similar cryptocurrency, blockchains, like Ripple coin, Dash coin etc., is basically for storing and transferring of value. They had very limited ability to program the transactions. And that limitation kind of prompted the idea of Ethereum and similar second generation blockchain, which is programmable via smart contracts.

We are now seeing what we call third generation blockchain and or enterprise blockchain. And enterprise blockchain, one of the early forerunners of the enterprise blockchain is Hyperledger. And then we will also see Corda very soon. And then soon there will be other blockchains, which are highly scalable with high concurrency. But let us focus now on Hyperledger.

(Refer Slide Time: 02:08)



But before we focus on the design choices of Hyperledger, let us understand this notion of order execute paradigm. So most smart contract based blockchains, they actually follow the idea of what is called a state machine replication. So state machine replication, we will go more into details of that, in this session or in the next session, is the idea that if I want a system to be fault tolerant, then I have to replicate the service.

So let us say I am providing a service, let us say a database service, but I want to make sure that if my service crashes for example, then I should be able to still able to serve the customer or serve the users. So what people have done and this is more than 40 years old theory from distributed computing literature is that you make replica of the see you can think of a service as a state machine, right.

So you start with an initial state, the user interacts with you, you change the state, and you then interacts with some more, and then you change to another state and so on and so forth. So you can think of a program or a service as a state machine. So if you want to serve customer's irrespective of possibility of, let us say crash, then what you would like to do is to have a replica of the same service.

But that replica must reflect all the state changes of the original machine, because otherwise suppose I crash at some point and then the user request is routed to the replica, if the replica is still in the initial state, then replica will not be able to serve correctly the customer and the customer it will be the user and the user will be very will understand that I am not any more talking to the service I was talking to.

So in order to make it transparent to the user as well as give availability to the service, you have to have the replica always be in lockstep with the original services state transitions. So therefore, you have to replicate what is called a state machine. And this is the idea of state machine replication. Now as you will see that other than crash, there are other possibilities. For example, a particular service may be down for maintenance.

So in that case, it is not a really a crash, it is actually a known declared you know downtime for one of the component of the service. Another possibility is that the either the original or the backup replica will actually start behaving weirdly because it has been taken over by a rogue entity or it has been affected by malware, or somehow its program starts behaving strange. And in such case, we say that that is a Byzantine fault.

So there are all these different kinds of fault models people thought about when they started talking about this replicating services. And we will see that, that is where the original problem of consensus came into existence. Then

you have to be in a consensus so that you are all agreeing on the next state for example. Now in case there is a fault, then also you have to keep maintaining this state transition as you interact with the user.

So this is called the active replication. So a protocol for consensus, or atomic broadcast needs to order the transactions that propagate to all the peers. So transactions may come to your system. And you have, let us say replicas, 4 replicas, or two replicas, whatever for your service. Now depending on where in the network this replicas are the transactions may come to the primary replica, or it may come to one of the secondary replicas.

Now as we saw, as we saw before, that if I give money from A to B, and then B tries to give money to C, then if B's original balance was zero, and the B to C transaction is only possible when A gave some money to B, then that ordering is very important. If the ordering is done differently, then in one case, both the transactions will succeed. In the other case, if B to C goes first, then it will not succeed, but A to B will succeed.

So as a result, if different replicas, do the transactions in different orders, then you will see the replicas are now out of state. Their states will differ. In order to avoid that, what you have to do is you have to ensure that the transactions are ordered and everybody agrees on that ordering. And if that ordering is agreed upon, then the state will also go in lockstep.

So in Ethereum, for example, what happens is, the consensus step decides that ordering, right. Remember that all of the miners are trying to create a block and in that block they choose the order of transactions and then everybody else, once they once the block has been actually created through proof of work or whatever the processes, then it is being sent to every other node and every node will eventually agree to add that as their blockchain next block and therefore, the state will always be the same.

But, one of the major issue is that ordering will never be different in different nodes, because the ordering is done by the winning the mining node. So consensus is done first in bitcoin in Ethereum and then once the consensus is done, then every other node falls in there in line with the same ordering.

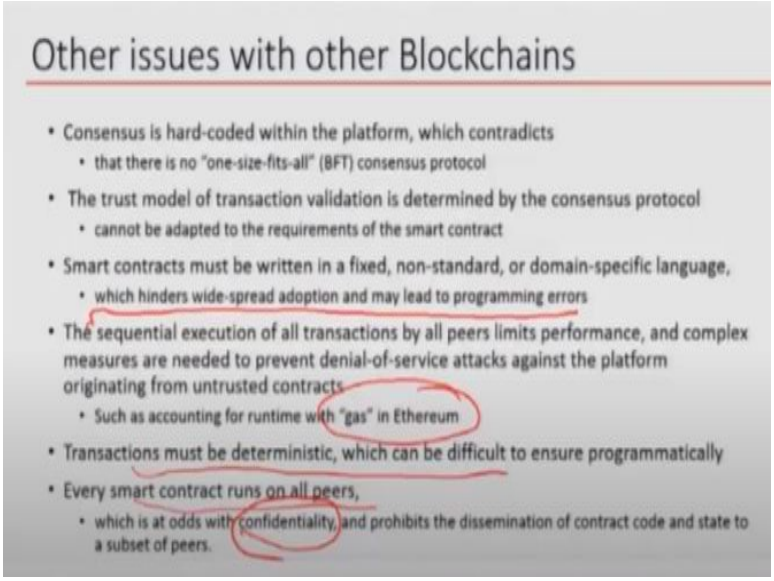
So therefore, we call it an order execute paradigm, which means that ordering is first and then execution. Execution means that every node will change their state in the same way, right. And then when the block goes to the other nodes, they will then do the transactions. They also have to remember they have to also run the same transactions and change the states and so on so forth.

So they have to do this sequentially as they get the block that wins. So this order execute architecture has been the first two generations of blockchain has been characterized by this order execute paradigm. And it requires all peers to execute the transactions and all transactions have to be deterministic. Remember that in Ethereum, we cannot have a random, call to a random number generator for example.

Because we do not want non-determinism because once the block has been decided, order has been decided, then that block goes to somebody else, they will also execute all the transactions and if there is non-deterministic execution, then the state in one node will be different from the state in the other node. So determinism is a must.

So order execute architecture is in Ethereum or in some other permission based ones like tender meat chain and Quorum. And then transactions must be deterministic. So these are the hallmarks of the earlier generation of blockchains.

(Refer Slide Time: 10:31)



The second issue is that remember, in bitcoin you cannot overnight change the consensus mechanism. Then you have to do some kind of a hard fork, right. Similarly, in Ethereum you cannot change the consensus mechanism, you have to do a hard fork. So therefore, the transaction we say that the consensus mechanism in this block chains are hard coded. You cannot change them and without creating forks.

In Hyperledger and more recent blockchain they have somehow extracted out the consensus mechanism so that you can actually depending on your need, you can change the consensus mechanism. So the trust model of transaction validation is determined by the consensus protocol. So trust model means what like so in case of bitcoin you do not trust any of the other nodes.

In case of Ethereum also you assume that nobody is trusted. And even these smart contracts are coming from anybody and everybody. They can actually deploy a smart contract. So the smart contract itself may not be actually, you know honest. And therefore, the trust model is that do not trust anybody. But in a permissioned blockchain since you have strong identity of each node or each user, therefore, you can decide what will your trust model be.

For example, as I was saying that if your blockchain has multiple organizations participating, then all the SBI let us say SBI, UBI they are together doing maintaining a blockchain and doing all their transactions on the blockchain. Then you can assume that the SBI employees they know each other because their identities are clear. And UBI employees know each other, their identities are clear.

So they can trust each other, trust within their group but may not trust across the group. So this is one type of trust model. Another type of trust model could be that you do not trust anybody. So you do not even trust your SBI colleagues, you do not trust your UBI colleagues. And then your mechanism for transaction validity etc. will be lot more complex. Because then now you cannot trust anybody.

But if you can trust somebody, then your mechanism for transaction whether the transaction is valid or not, etc. will be different. So now this trust model is different from the consensus trust model. The consensus trust model is about what is the fault the what is the fault model you are assuming about the consensus mechanism. So the remember the consensus mechanism is about ordering the transactions.

And so if you assume that the nodes that are participating in the consensus mechanism, some of them can be rogue or some of them could be affected by let us say malware, then you have to have a very strong consensus mechanism. If you trust all these nodes, then you can have a very simple consensus mechanism. So therefore, what you have to do is that, you have to consider two different trust models.

One model is about those who are doing transaction among themselves, how much they trust each other. And another trust model is that those nodes which are autonomous and which are doing the consensus, how much you can trust those. And in bitcoin and Ethereum these two trust models are intermingled. There is no difference in these two trust models.

But in a permission scenario, this can be very different and therefore, we have to somehow extricate the one from the other and have two different trust models and you can play with the trust model. You may have, if you have a very

nice you know secure set of nodes which are doing consensus, then you can fully trust that. Whereas, since the transactions are coming from outside from clients and so on, you may not trust.

Or you may see that the all the people who are doing transaction are within your organization, you trust them. You can have a good trust, but you may say my consensus mechanism is maybe faulty. Some nodes may actually crash and so on. So I cannot trust that fully. So I have to have the right complex consensus mechanism. So you can have all kinds of mixes of the two trust models.

So other thing is that in bitcoin for example, you have to write any kind of complex logic, business logic like micro payments or escrow etc. in a nonstandard fixed language called bitcoin script language. In the case of Ethereum, you have to write them in Solidity. Now there are some few other languages which can also converted to the EVM bytecode.

But, in case of Hyperledger as I said before, you can write the smart contracts in any programming language, because there is no EVM concept or VM concept associated with them. You are basically going to execute the smart contracts inside dockers and in a regular programming environment. And this fixation with specific language is, you know hindering the wide adoption of this blockchain technology.

Because there are not many people knew Solidity, not many people who knew Go, but lots and lots of people know Java, C++, Python, etc. So you can do that. Now another issue is that in bitcoin and Ethereum etc., all once you get a block, you have to execute all the transactions in that block, even though you did not mind that block.

And so therefore, if a transaction, if a smart contract that you are executing is very time consuming, then you might get into an infinite loop or very long loop. And then that particular node will become engaged for a long time, and that can lead to what is called a denial of service attack. That node will not be able to add to the blockchain and therefore, nodes will be busy processing the last block's transactions.

So in Ethereum this is solved by using gas mechanism, right. But in the gas, mechanism works because Ethereum has a native cryptocurrency. So you can talk about we that is the in a smallest unit of Ethereum and you can say gas costs so many weights and so on so forth and you can do that. But if you are having a permissioned blockchain in a private setting, you do not have a native cryptocurrency.

Therefore, you cannot really have this notion of gas or or incentive mechanism like in bitcoin and therefore, you cannot stop this possibilities of infinite loop and so on. And so therefore, this you cannot afford to run the smart contracts at every node. Smart contracts are only run on specific nodes, which you trust. And the smart contracts are written by specific entities that you trust.

So therefore, you assume that they will not put in an infinite loop or some kind of a very computationally intensive logic inside the smart contract. So because you are trusting so much you can afford to do that. Therefore, that is why you are writing programs in general purpose language C, C++, with which you can do such things like creating infinite loops very easily.

But you assume that smart contracts are not written by random untrusted people. It is written for specific business purpose by the entities that you know very well. Their signature is associated with the smart contract and the corresponding to that signature there is a digital identity and so on. So therefore, you can do afford to do that.

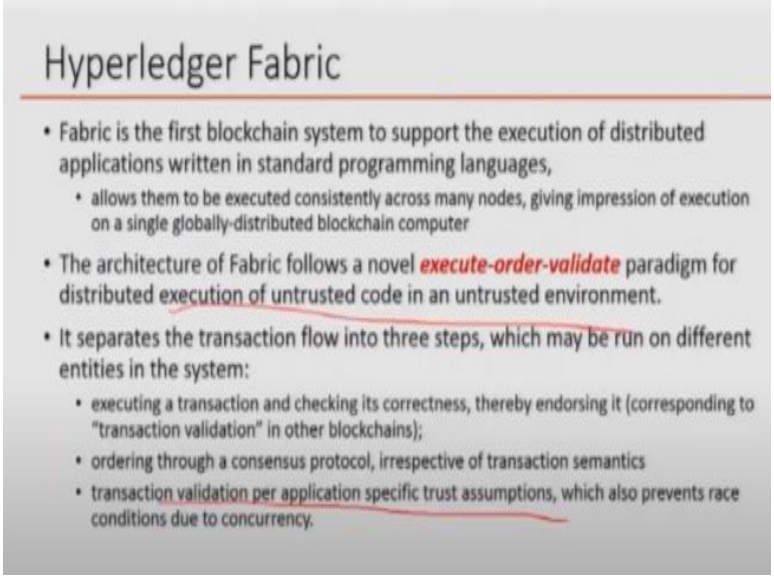
Also in those platforms like Ethereum therefore, remember that otherwise there will be divergence if the smart contracts have non-deterministic logic, because once the block comes to you being mined by one of the nodes, you have to execute the transaction. If the transaction was non-deterministic, the result you will get will be different from the other blocks what the other nodes will get, and then there will be a fork in the blockchain.

And this will be too often and therefore, you cannot afford that. So determinism is has to be inbuilt into the language in which smart contract is written. And to do that your language should be designed and that is why there is specific languages like Solidity or Ethereum you know scripting language, which do not have any construct for non-determinism.

And then, again, this we talked about earlier that every smart contract runs on every node in case of Ethereum, which means that every node needs to know your smart contract the byte code of your smart contract. And therefore, any intellectual property you have in the logic of the smart contract or any business secret as to how you process transactions, how you make decisions on the transactions etc., will be known to everybody else.

That is also not desirable. And therefore, the confidentiality of the business secrets etc., is not kept and therefore, we need to have some way of also not having a system in which smart contracts will be run at every node. So all these things that they figured out in the earlier blockchains that they want to get, make a difference from.

(Refer Slide Time: 20:58)



So Hyperledger has many different incarnations. So Hyperledger Fabric is the one of the most used one. So we are talking about Hyperledger Fabric. There are other implementations of Hyperledger, but Fabric is the, I would say most canonical implementation of Hyperledger, which is done by IBM. So Fabric is the first blockchain system. So to support execution of distributed applications written in any programming language.

So this gives us the ability to attract people to use it. So there is adoptability will increase because, if I have to code a business logic in C, I already know C, I do not have to learn Go or Solidity or whatever that makes the barrier to entry into this technology much lower. So the architecture of Fabric follows execute-order-validate. So earlier we were talking about order execute.

So order first execute next. Now we are talking about execute-order-validate paradigm. And we will explain what that means for distributed execution of untrusted code in an untrusted environment. So it has a transaction flow, which has three steps. First executing a transaction and checking its correctness and thereby endorsing it. So there is a step called endorsement.

So when client or the user of the application proposes a transaction, for example, give this much money from my account to somebody else's account or give this much price for the entity I am purchasing from that other entity. Then, the endorsers are certain nodes which will simulate the execution of that transaction and say If the transaction happens, then this will be the final situation.

Your account will be debited, that account will be credited or things like that. So that information along with an endorsement which is which comes in the form of a digital signature of the endorser comes back to the client or to

that code that is running on behalf of the client. Then, after and this endorsement may be from one node, or it may be from multiple nodes depending on what is called an endorsement policy.

The endorsement policy is also flexible. So then, once you have got enough endorsements, as per your endorsement policy, if the endorsement policy says one endorsement is enough, that is fine. That then you are done with one endorsement if it says that it has to be a two out of these three nodes, then that also has to be gotten, things like that.

And then what you will do is that you send the send this endorsements with the possible changes in the state that is how much your balance will be, what will be deducted from your balance, what will be added to the other balance etc. to the ordering service. Ordering service will then order the transaction. Ordering service does not care about the semantics of the transaction, it does not execute the transaction.

It just do the check the read/write sets and order them. And there is some notion of a version issue also. But right now let us not worry about it. Just understand this that the transactions will be ordered and a block will be cut and that block will be sent to peers called validating peers. So then validation will be done by this validating peers.

What they will do is they will check which transaction is meaningful, which transaction is meaningless and they will mark those transactions which are meaningless. And for the rest of the transactions, it will use the read/write set to change the state replica of the entire system in its local copy and add that block to the end of the blockchain. So this is validating peers are also designated peers, not everybody in the network has to do validation.

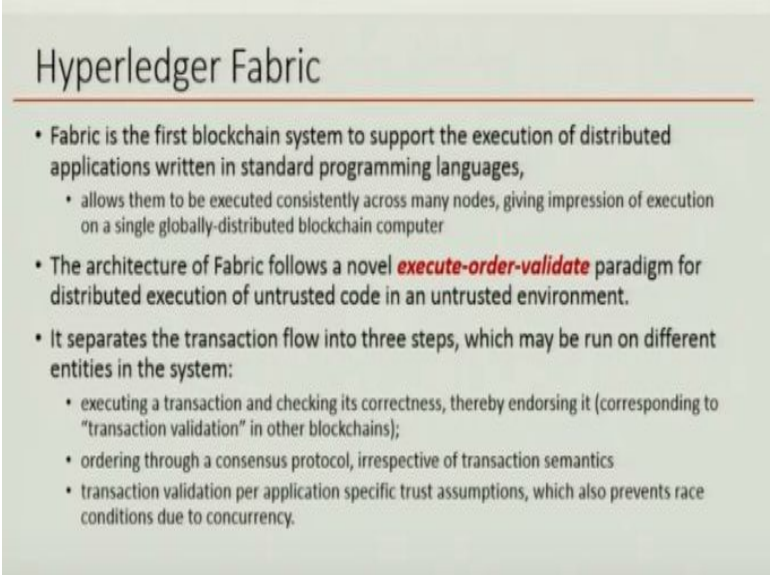
So you see that for a transaction, not everybody executes the transaction or everybody does not need to endorse the transaction for each transaction or application. There is a transaction endorsement policy and that has to be fulfilled. Similarly, validation also is done by specific validating nodes. And when we come back, we will get into more details of Hyperledger Fabric.

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

Lecture - 22

Welcome to another session of Blockchain Technology and Application on NPTEL. So last time in the last session, I was talking about Hyperledger Fabric.

(Refer Slide Time: 00:25)



And I was talking about this execute-order-validate paradigm as compared to order-execute paradigm in the previous blockchains that you have seen. So going back a little bit. So Fabric actually distinguishes itself from the previous blockchains that you have studied, like bitcoin blockchain or Ethereum blockchain is that it follows the execute-order-validate paradigm and the transaction flow has three steps.

First executing the transaction and checking its correctness. And this execution does not change anything in the blockchain or in the state of the blockchain. It just in a sense simulates the execution of the transaction. And then the transaction results are endorsed by some of the nodes. They are called the endorsers, and then endorsers send the result of the, of executing the transaction as read/write sets.

And these read/write sets are then sent back to the client which originally requested the or proposed the transaction. And the endorsement is also given by multiple different nodes based on an endorsement policy. We will see what endorsement policy means. And each endorser also attaches the digital signature of that node so that others can check whether the right endorsers have endorsed the transaction.

Then the transaction when the enough endorsements are collected by the original client, then it sends it or broadcasts it to the nodes that are responsible for consensus. And here the consensus basically what it does is that it orders the transactions. And the ordering of the transaction is based on this read/write sets and ordering does not care what these transactions are about what kind of domain it is being done on.

Like for example, it could be financial transactions, it could be land record transactions, it could be other kinds of any other kind of transaction about supply chain, it does not care. So transaction semantics is not looked into by the ordering nodes. The ordering nodes basically run a consensus algorithm and then they do decide on an order.

And then once they have ordered the transactions, then they will bunch them together from various clients multiple transactions will come. If these transactions are working on the same data then the ordering is more important. If the transactions are on completely different sets of data, then their ordering is not that important, but in any case the orderer will order the transactions. And then this ordered transactions are put into a block.

And we call it cutting a block and then the block hash is computed. A sequence number is given to the block and then the block is broadcast by the consensus nodes to the rest of the nodes. The rest of the nodes, not necessarily all the nodes depending on what the administrator of the blockchain decided who should be the nodes that are called validator nodes.

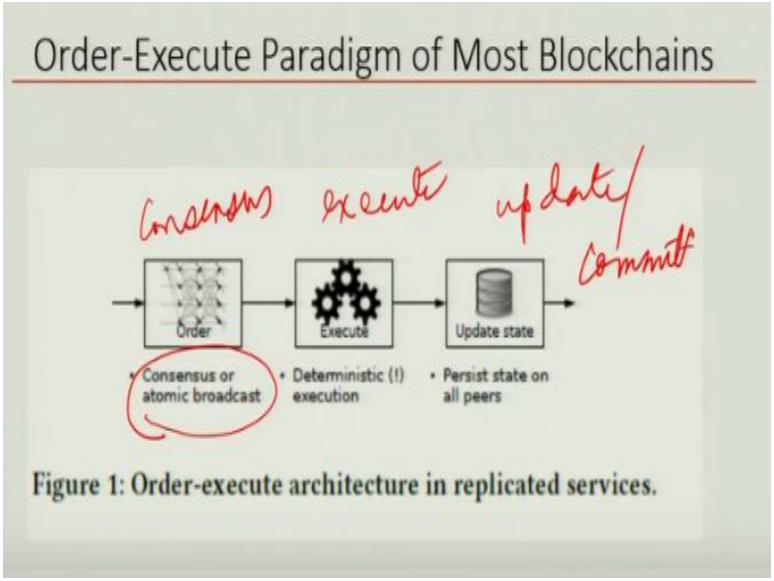
The validator nodes will look into every transaction and then look at the version information in the read/write set and decide which of the transactions are valid and which transactions are not valid. For example, if the version number of a particular data being read in a transaction does not match with the version number of the same piece of data in the state database, then that is not a valid transaction.

Because that means it is reading a different version value of the value of that particular piece of data. And therefore it will if you execute that transaction, it will be bad. So those invalid transactions are then marked. And this marking is done by creating a bit vector, such that valid transactions are set to be 1 and the invalid transactions are set to be 0. Since the transactions are ordered we can do a bit vector to mark it.

And then everybody, every node that is maintaining the blockchain will add that node to the end of the current blockchain and therefore, and then update the state for the valid transactions. The valid transactions will write may write on certain nodes, a certain date piece of data. So that will be written in the state database. So that is how the execution ordering transaction execution ordering validation happens.

And that is the end of a particular block activity. In the meantime, the clients might have submitted more transactions for endorsement. Endorsement might have come and then they were sent to the orderer. So these things are happening sort of in a pipeline. So that is what is the execute order validate paradigm means.

(Refer Slide Time: 05:59)

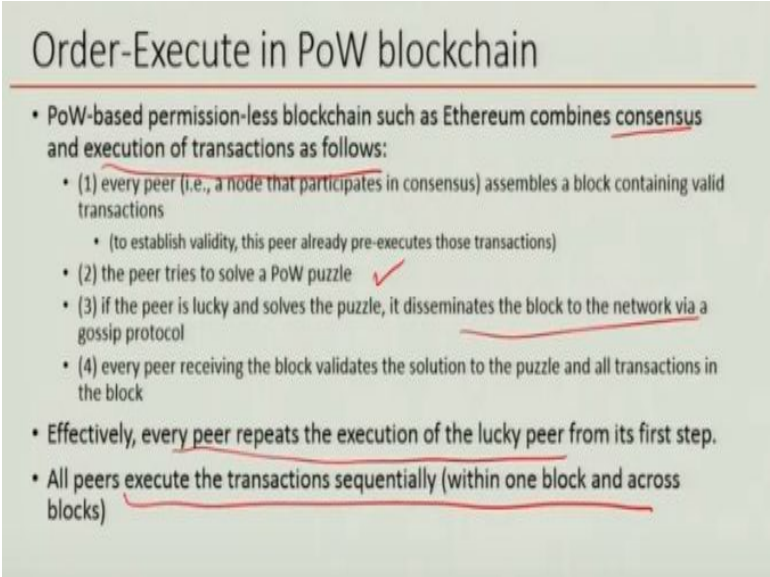


So going back pictorially if you want to see what this order execute paradigm is, which is first a consensus happens. So everybody gets broadcasted all the transactions. I am talking about bitcoin, Ethereum, etc. So everybody gets a copy of any transaction that happens, then they will bunch some of these transactions together and they will try to cut a block, but in order to cut a block, they have to solve a puzzle.

And once they solve the puzzle, then they have won that round of consensus. Then once the consensus is reached, the winning block is broadcast to everybody. And then everybody has to then execute the transactions at their own nodes and make sure that these transactions are valid, they have the right signatures, etc. And then they will update the state and add that block to the end of the blockchain.

So this is the consensus is ordering. This is the execute and then update or commit. You commit the those set of transactions. So that is what we saw before.

(Refer Slide Time: 07:23)



So now we are we explained to you that we do it differently in Hyperledger. Now let us see, let us discuss a little bit on order execute in proof of work block chains like bitcoin or Ethereum. So we combine the consensus and execution of the transaction as follows. So every peer assembles a block containing valid transaction. To establish the validity, you already pre-execute the transactions.

You execute the transaction and you only execute the transactions if the signature matches and the output the you know coin is going to the going to a valid public key etc. The peer then tries to solve the proof of work puzzle. And if the peer is lucky, and win the puzzle, it disseminates a block to the network via gossip protocol. And every peer receiving the block validates the solution to the puzzle and all the transactions in the block.

And doing so they have to also execute the transaction. Every peer repeats the execution of the lucky peer from its first step. And that means that we are doing this execution again and again and again in all the nodes in the system to although it is happening concurrently, but within each node, the execution of transactions are happening in a sequential manner.

But if you think back about what I said about Hyperledger, I have a transaction I am getting endorsed. You have a transaction you are getting endorsed, maybe by the same set of endorsers, maybe a different set of endorsers. So I can have the execution of multiple transactions in parallel. Because if my endorsers do not overlap, then those transactions can altogether be endorsed and sent back to the respective clients.

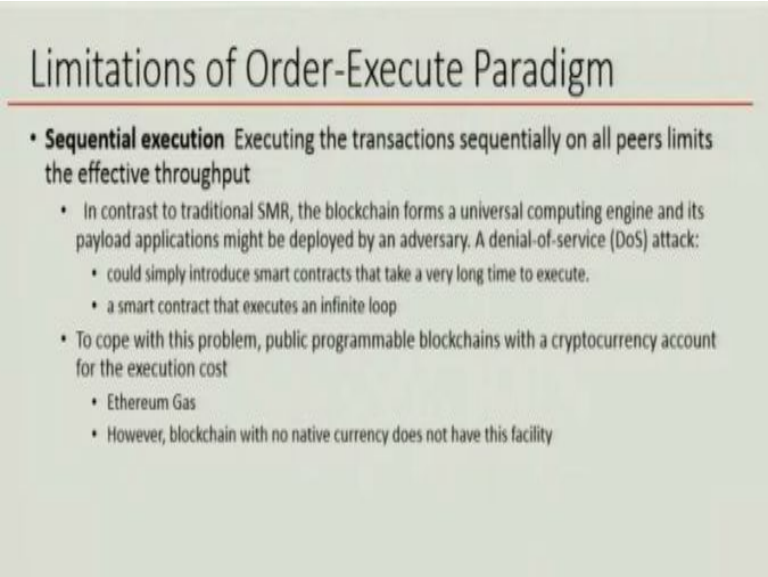
And all the clients can then submit those transactions to the ordering service. So therefore, and then afterwards, we do not execute the transactions, we only use the read/write set to update the state if the transaction is valid. So therefore

execution happens before ordering and before validation. And these executions can also happen in parallel in concurrently on behalf of multiple different clients provided the endorsers are not all common.

So that is the difference that throughput for blockchain that are order-execute based their execution of all the transactions even though those transactions came from different clients into the system, they are all executed once a block has won, that block's transactions are all executed sequentially. Whereas here in Hyperledger, I can have parallel execution of transactions.

And then I do ordering and then I do the validation. And then after validation, everybody updates their state. So that is one advantage that Hyperledger has in increasing the throughput of the transactions.

(Refer Slide Time: 10:48)



So as I already hinted, that this can be viewed as a limitation of the older order execute paradigm, the transaction's sequential execution by every peer after the block the POW problem has been solved is bad for the effective throughput. Now POW itself takes time as we know and then on top of that everybody has to execute this transaction sequentially.

So therefore, the other problem that can happen is among these transactions remember these transactions are being set as in case of bitcoin through bitcoin scripts. And remember the transaction execution would mean that I put the scriptPubKey and ScriptSig of the previous execution, previous transaction associated with this transaction's inputs together and execute that.

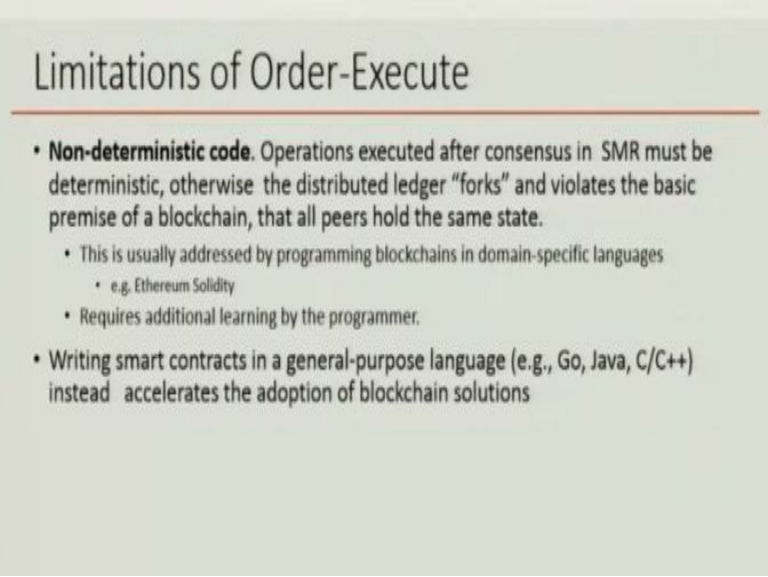
So each transaction execution is basically executing a small program. In case of Ethereum each transaction would be executing a function or multiple functions in the smart contract and therefore, if this smart contract is written in bitcoin, this is not easy to do or not even possible to do because you cannot have loops in a bitcoin script.

But in case of Ethereum, what you will you could do is that if you are a bad player, you could put a smart contract and convince everybody to execute a function in that smart contract in and the function itself may have an infinite loop or whatever. And therefore, you can launch a denial of service attack. Of course, Ethereum solves it by limiting the gas etc.

But if you do not have a native cryptocurrency, in your blockchain, then how do you price the gas, right. So therefore, it is a free for all. You can actually make the system execute very long functions and therefore slowing down the throughput very much. So Hyperledger does not have a native cryptocurrency. And Hyperledger is even allowing writing smart contracts that are not even in a stylized language like Solidity.

It is it could be C, C++ whatever, where you know very long loops can be possible and there is no notion of gas or gas limits. So therefore, executing a transaction sequentially by all nodes would not be a good idea, because very easily one can do a DOS or denial of service attack. So that is the idea, one of the limitations of order-execute paradigm.

(Refer Slide Time: 13:46)



Another issue, we are already hinted about that in the last session is that you cannot have non-deterministic programs as smart contracts in the Ethereum or non-determinism in the bitcoin because if the transaction execution, non-determinism means that the same code gives you different result at different times, right. For example, the function if you call random function twice in two different nodes, it will give you different results.

So therefore, if you want the blockchain not to get forked because the value the result of a transaction is different in one block and they are different in another block then you must make sure that your smart contracts or scripts are deterministic. So to do that, to ensure that what you do is you create special languages in which there is no way to create non-determinism right.

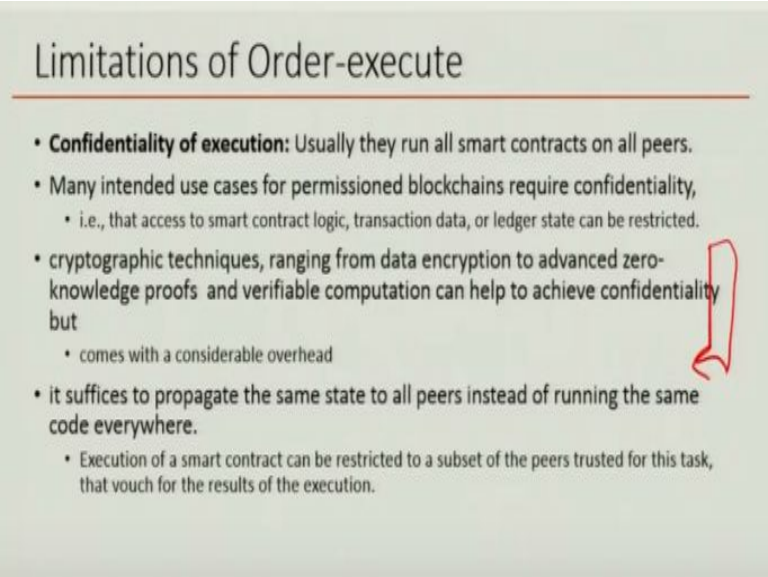
So now you might say that if I limit my language will it still be Turing-complete and turns out yes, it will be still be Turing-complete and but you do a domain specific language and then force people who will write smart contracts to program in that language. Now when you want to bring a new now there are a lot of solidity programmers because Ethereum got popular.

But if you start a new blockchain like Hyperledger and say that you have to program it in a very specific language, then there may not be any takers and therefore, the blockchain may not become popular and it will die. So therefore, you have to allow anybody who can program in any language to be able to write contracts. And therefore, and then if you do that, you cannot stop non-determinism because those languages will have non-determinism as a built-in property.

So therefore, what Hyperledger decided is that they will not worry about non-determinism because their final result does not depend on determinism and non-determinism because if the transaction is non-deterministic, then during the execution itself right, by the endorsers, when the endorsements come back, you will see different the client will see different results coming from different endorsers the read/write sets and this transaction will be discarded.

So therefore, we do not have to limit the language and still have determinism in the blockchain state.

(Refer Slide Time: 16:37)



Another issue is confidentiality of your IP. So suppose you are using Ethereum and you are creating distributed application based on multiple smart contracts which are meant for supply chain management. Now Ethereum has, you know other nodes which are not interested in your application but they will all get all your smart contracts because all smart contracts are part of the blockchain.

And so they can see how you have what your business logic is. If they you know take the byte code, do reverse engineering, they can see all that. They will also see the results of all your transactions. In fact, they will execute all the smart contracts. What that gives you is that, so in such a system in a public blockchain, you cannot keep confidentiality of your IP. The IP means intellectual property the way you run your application.

So that basically means that if you want confidentiality, then you have to have in such that your smart contracts are not part of every node or it is not part of the blockchain. So who has the smart contracts in them, the endorsing nodes. They execute the transaction, so they have to execute the smart contracts. So if you trust the endorsers, because they are part of your business environment or business entity, and those who run those nodes are also having real world identity associated with their identity.

So therefore, you can have the endorsing node know your chain code, which is the smart contract in Hyperledger, but no other nodes the orderers do not need to know your smart contracts. The other nodes that are keeping the blockchain do not need to know your smart contract.

And Hyperledger was specifically thought about as a way for multiple businesses to work together on workflows and data sharing that is meant for inter business activities. Supply chain management is an inter business activity. Banking

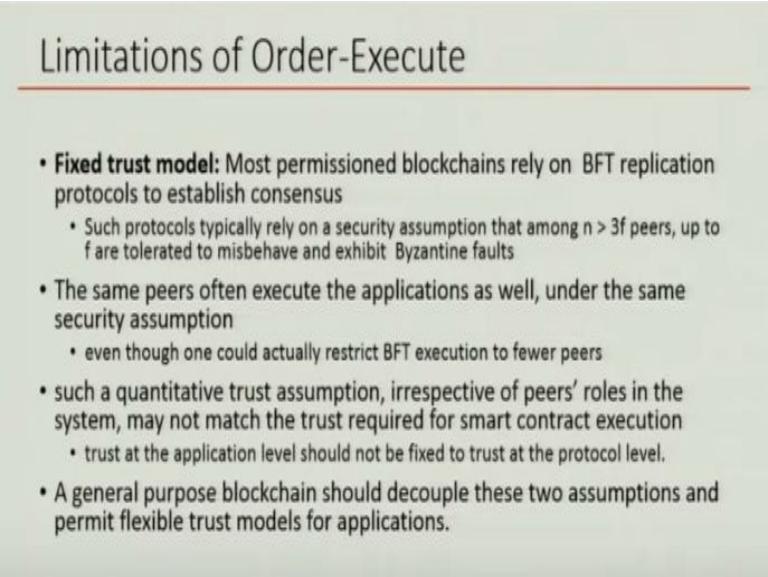
between multiple banks is an inter business activity. So you may trust all the nodes that are associated with your bank, but you may not trust the nodes that are associated with another bank.

So these kind of environments will be supported, if you have this, cannot be supported on say Ethereum but you can support this on Hyperledger. Now you might say that well in I can do lot of, you know very complex use of cryptography, zero knowledge proof etc., to ensure that kind of thing the confidentiality and probably yes. But it will have a lot of overhead of running cryptography, running zero knowledge proof which basically has heavyweight cryptography.

So therefore, that is not a very good solution. So what Hyperledger decided that all you need to know for all you all the every entity needs to know who is the purveyor of your business blockchain is the state, what state be after every transaction, the system goes into. And what transactions were done, but the transaction ID, etc., who did the transaction, all that stuff.

You do not need to really have the transaction details or the code for the smart contract and every node. So that is the basic idea.

(Refer Slide Time: 20:29)



The other thing that again, we discussed, you know superficially last time, the trust model. So blockchain in bitcoin for example, you do not trust any of your peers. All the nodes are untrusted and therefore, you have the consensus algorithm, which is very complex, very expensive, with proof of work etc. In Ethereum again the same thing. Now in a private version of an Ethereum, which is not public, you can simplify the consensus process if you trust most of your nodes.

And then you can use something like a proof of authority proof of you know other kinds of consensus techniques. But in a public blockchain, your trust is with no, you do not trust anybody. So that is the trust about the consensus, like how the consensus making nodes are trusted among each other. That is one type of trust. The other type of trust is the trust of the business application, right?

So do you trust everybody to faithfully execute your transaction, everybody to not breach the privacy of your code, not breach the privacy of your activities, etc. So what Hyperledger decided is to split this notion of trust of the business application and notion of trust in the infrastructure. The infrastructure, which is responsible for consensus, there the nodes may all trust each other.

Or the nodes may have some assumptions about trust, like we only we trust everybody, but some of the nodes may be compromised. So we only trust maybe one-third of the nodes, two-thirds of the nodes, etc. And accordingly, you can design your consensus algorithm. But the trust mechanism at the business level would be enforced by a notion of what they call channels.

So what the channels do is that they separate the on the same infrastructure you can run multiple blockchains. So one entity might be running a supply chain application, and one the same entity can run the supply chain application and some entity can run the banking application or monetary transaction applications between the same set of businesses.

And they do not need to see each other let us say or they do not trust each other because these are two different entities. One may be audited another one may be for internal purposes etc. So you have this notion of channels. So the channels basically maintain separate block chains. Every channel has a separate blockchain, same set of nodes are running.

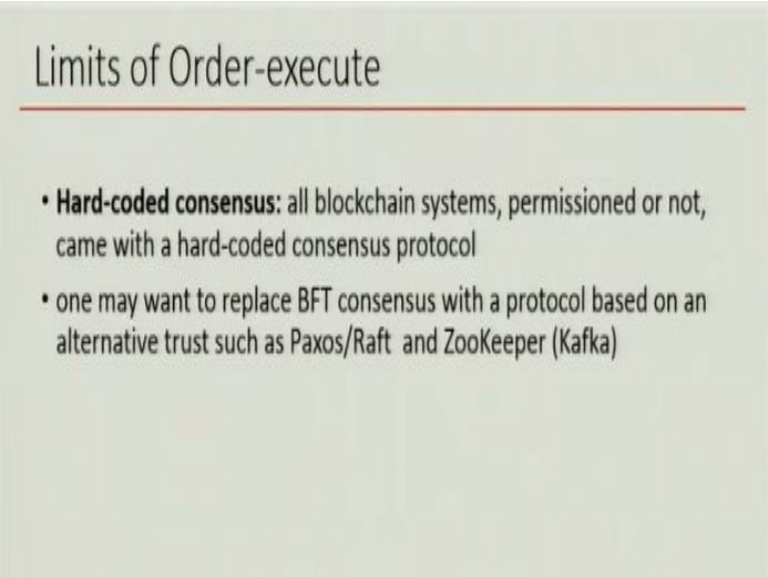
The contracts, smart contracts associated with each channel are also different but they are running on the same node, but on separate dockers. So they are completely separated. But the ordering service which is transaction agnostic, semantics of agnostic, they do not care what business it is and so on. They just look at the read/write set the version numbers etc., and do the ordering.

So they can be common between the two because you are never sending your code or you know you do a semantics of your data to them, you are only asking them to do the ordering. Now that set of nodes which are doing ordering has a

business semantics agnostic infrastructure activity, which is to order transactions. So they can order transactions for this application, they can order transactions for that application, it does not matter.

So that split of trust is possible because your execution is never happening by the same nodes who do consensus. So that is the other idea that is not possible in the order execute paradigm.

(Refer Slide Time: 24:45)

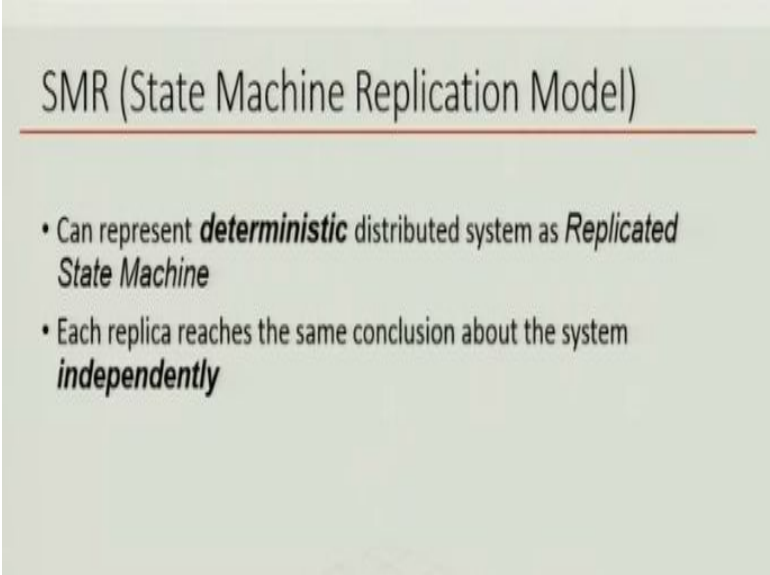


And the final thing is that the in bitcoin and blockchain the this thing is hard-coded, the order-execute blockchains they have the consensus algorithm hardcoded. Which means that bitcoin has proof of work, Ethereum has proof of work, but if it wants to go to proof of stake, it has to do a hard fork.

So whereas here since you your ordering service or consensus service is agnostic of the business application, therefore you can decide the system configuration person or the blockchain configurer may decide that I will use simpler consensus mechanism if I trust all the nodes that are involved in consensus. If I feel that there I need crash tolerance, I will use a different type of consensus mechanism.

If I need Byzantine tolerance, then I will use a different type of consensus. If I trust if I do not worry about crash and Byzantine behavior then I can even use a single node which will do all the ordering, which is called a solo. So that also is this consensus is pluggable.

(Refer Slide Time: 26:09)



So now let us talk about this state machine replication model. Because blockchain as I have mentioned the most of the technology that is used in blockchain is reincarnation of many things that the cryptography community, digital cash community and distributed algorithm and distributed computing community has invented over last 40 or more years.

So replicated state machine is one such model for distributed computing. And then the idea is that if you have a database server, and that database server is actually in the database server, you have you have a dedicated service that accepts your transactions and then execute them on the database and then sends you back the results or success or failure etc. So that service, if it is a single service, then it may crash.

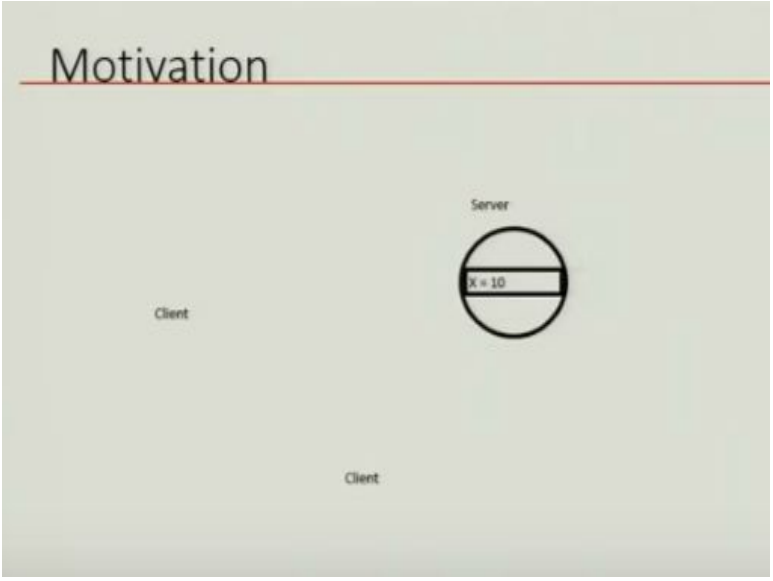
So therefore, people have started thinking about primary backup etc. So primary backup would be that there will be a primary for the service and there will be a replica for the service and the replica will be having the same set state transitions as state changes happen in the original primary service, because, if suddenly it has to take over from the primary because the primary crashed, then it has to be in the right state to continue as if nothing has happened.

So therefore, the question is how do you make sure that one or more replica are in the same state and every time they decide to do the same activity all the time. And this is where the original problem of consensus came about.

So each replica and you cannot have a leader follower thing all the time, because you have to then also elect a leader right, which basically is another way another form of consensus mechanism, that who would be the leader and who would be follower to follow the commands. So therefore, it is best that there should be a way that everybody independently come to the same conclusion about something.

It may be that about choosing the leader and then follow the leader all through, or it may be that in every step, you want to build a consensus about what to do next, or what state to go to next, and then go there everybody independently goes there. But there has to be a consensus.

(Refer Slide Time: 29:03)



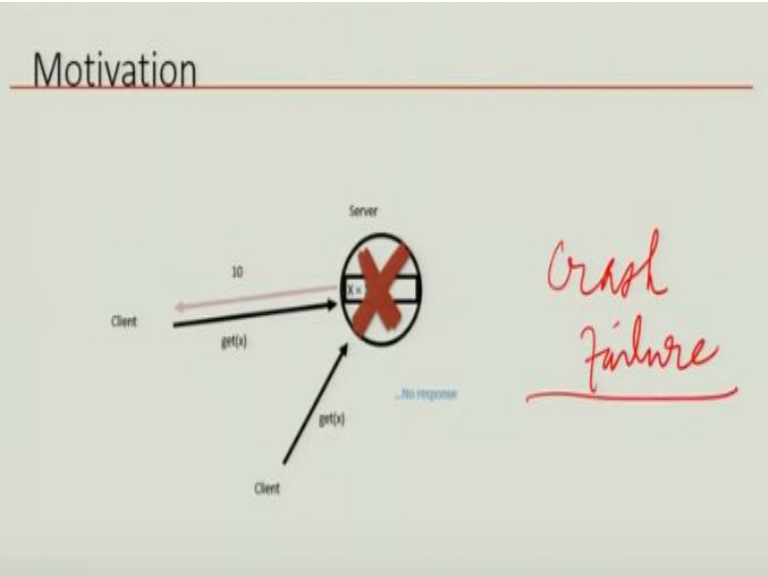
So let us look at some simple animations to understand this. So what but before that, we will take a break, and then when I come back, we will start with the animations. So we will see you next time.

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

Lecture - 23

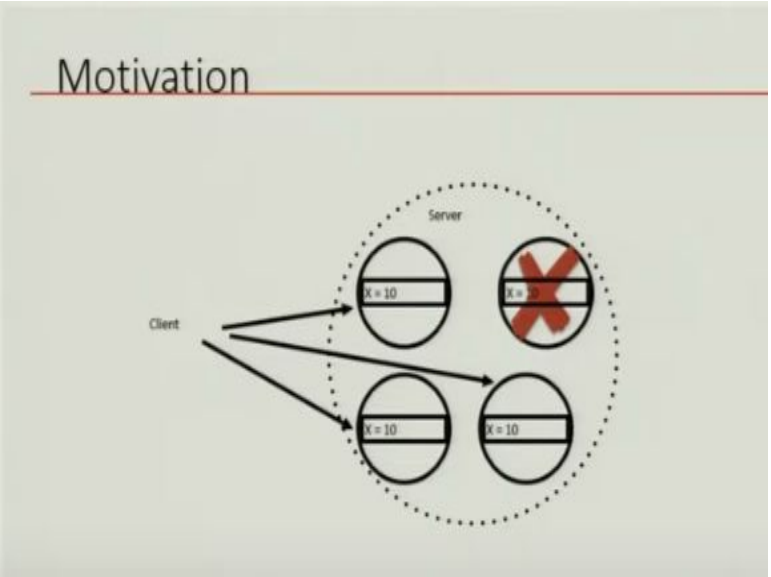
Welcome back. So we were talking about state machine replication. And we wanted to show you some animation to make you understand about state machine replication. So let us say I have a server.

(Refer Slide Time: 00:29)



And the server maintains some data or state. And let us say currently the value of some variable x is 10 in the server. Now suppose a client says what is the value of x now right? So you will tell them that the value is 10. Now let us say the another client says that change the value but at this point, the server crashes. So at this point, if it says give me the value of x, then you will not get any response. So this is what is called a crash failure.

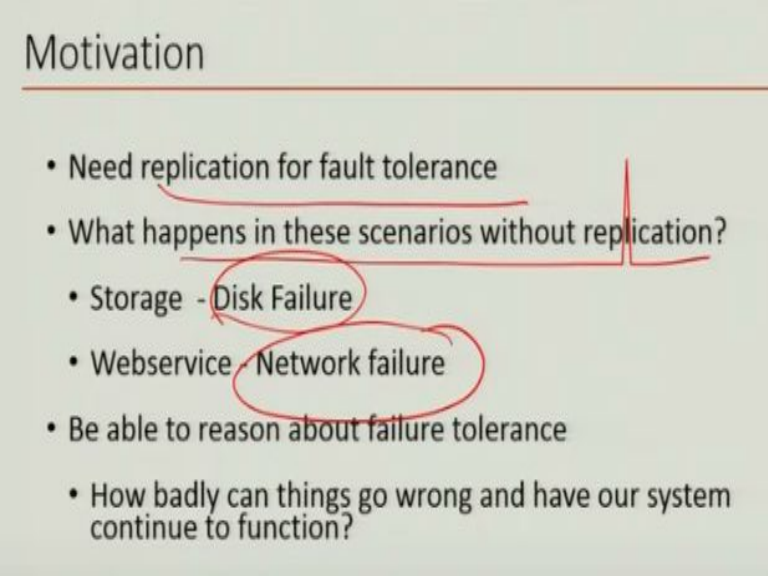
(Refer Slide Time: 01:22)



Now the question now is that in order to avoid such a no response scenario, what people started doing is to have replication of state machine. So instead of one I will have three other machine which are exactly replicating every state transition of the original one, okay. So you would think that would solve the problem. But it will bring other problems. So let us see what kind of problems it may have.

So let us say one of the four has crashed and three are alive. And then the client asks something. Client does not know how many there are right? So client is asking this is for to the client, this is a service, right. Now whether the service is implemented as a replicated service or as a single service, the client does not care. So the client is asking and it will then be forwarded to all the replicas. In this case, one of the replica is dead. So it cannot be sent to that.

(Refer Slide Time: 02:28)

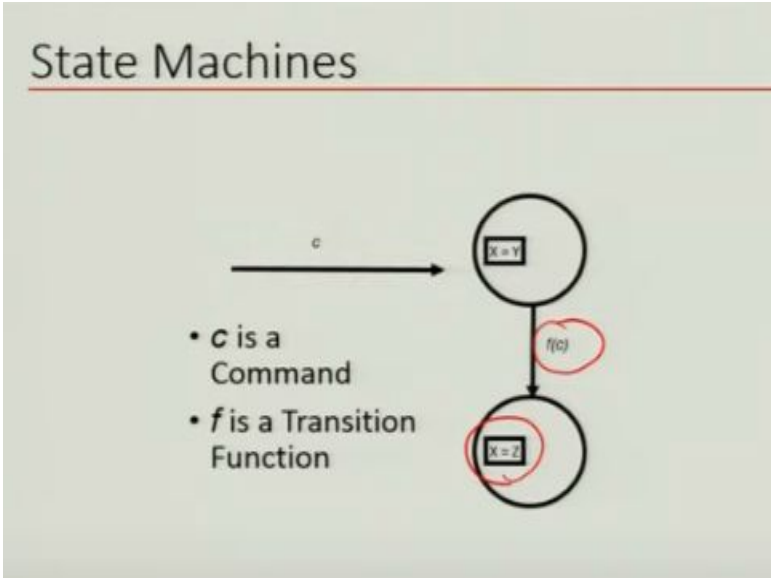


So this is what is called a replication for fault tolerance. So what kind of you know if you do not have replication, what could have happened? So first of all, you can have disk failure and in case of service is over the network, then

you have may have network failure. So you have to be able to reason about failure tolerance, which means that if one of the replica fails or more replica fails when can I guarantee that the service will still work correctly as far as the client is concerned?

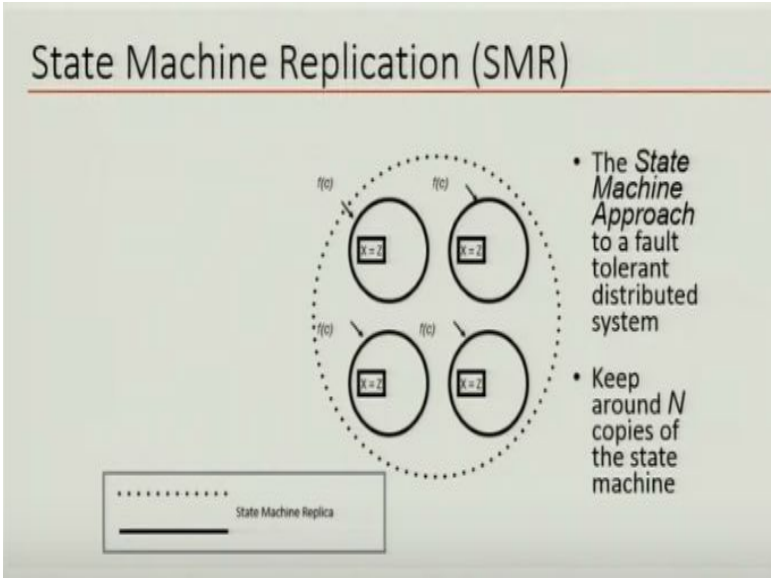
And we will see what can go wrong? And when can we not guarantee that the client will be it will be totally transparent to the client, whether something has failed or not, how many failed etc., can we guarantee that?

(Refer Slide Time: 03:29)



So let us say I have a state machine and I have a client. And client send some command, right? And that command will lead to a state change, right? For example, if this command is basically saying, set the value of x to a new value z, then a state change will occur because now the value of x has now changed from y to z. So it is a new state. So f is the transition function, it does not matter what you call it.

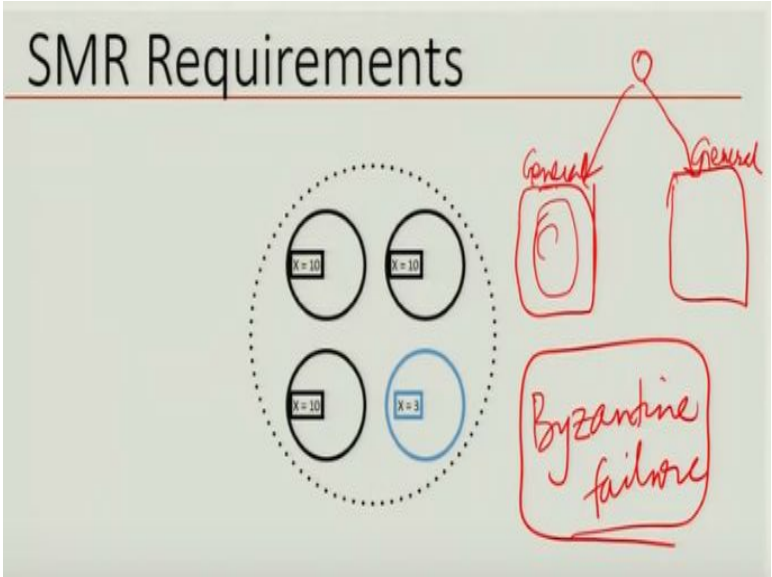
(Refer Slide Time: 04:06)



So now suppose we have this situation. For earlier picture, we saw that we have only one server, and it is changing state. So this is a state machine. Now let us say you have four state machines, and they are replicating each other, which means that they are always having the same, they must be having always the same transition unless one of them or more of them have failed.

Obviously, then they cannot do any transition. So let us say I have four copies. Now let us say I send the command. So it goes to all of them as a state change activity.

(Refer Slide Time: 04:49)



And the state of all of them will change. So now let us look at this scenario where x is three in each of the replica. So they are in the same state. Now you send a command, which says that change the value of x to 10. So if it is replicating state machines, then everybody should change it to 10. So under normal circumstances when nothing fails, etc., that is what you expect. So this is good.

Now let us consider another scenario, when the same situation happens, you want to change x to 10. And that happens. So that is good. Now you want to say x equals 10. And at this time, something goes wrong in the one of the servers, and it either is unable to change the state or it becomes malicious. And that kind of malicious behavior when it acts on its own will, rather than acting as per instruction then we say that, that machine is showing what is called a Byzantine failure.

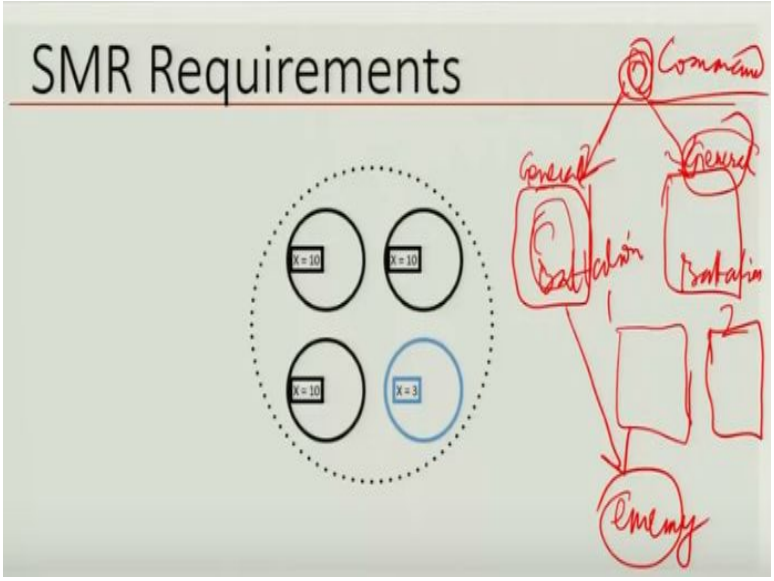
So this is Byzantine failure actually encapsulates many different types of behavior. For example, pretending to be dead could be a Byzantine behavior. Actually dying would could be interpreted by as a Byzantine behavior. Or

behaving differently then how it is programmed, see all these state machines have been programmed to follow each other.

But, if one of the machine decides not to follow, then it will be also Byzantine behavior. And then when it does not follow what it will do, we do not know. It can be do arbitrary random things, right. So why is it called a Byzantine behavior is a long story. So there is this thought experiment. In this thought experiment, you have a commander and you have two sets of army.

And the commander is supposed to send command. And then these set of this, the general here and general here must are supposed to obey the order. For example, if the commander says, both should attack.

(Refer Slide Time: 07:36)



So let us say I have a situation where the enemy is here. And you have battalion 1and battalion 2. And commander is going to decide strategic based on his strategy, his knowledge of enemy position and his knowledge of whether one battalion is enough to you know win over the army or both the battalion is needed and accordingly he will give command.

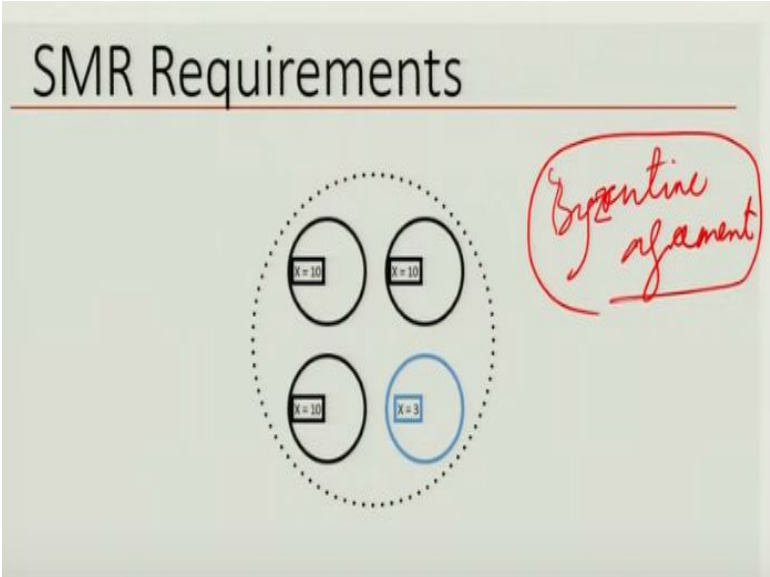
So let us say the commander decides that both the battalion should go and attack otherwise the one battalion if it goes it will be decimated. So if the commanders sends messenger to both saying that attack. Now if this general is unfaithful, knowing that the order is attack, he might say, I got the order to retreat. So this general will not know that so he will attack and get decimated.

So the question is that how can the commander ensure given that there is a possibility that one of his general would actually not behave properly. How can he assure that there will be enough battalion to actually win, right? So he cannot do this even with three generals. In fact, if there is at least if he assumes that only one general will defect, even then he will need four battalions and four generals.

Out of which only one can be Byzantinely behavior in a Byzantine way, then only he can ensure that he can win, right. So that thought experiment was done by distributed algorithms people back in 70s to actually formulate this notion of fault tolerance. And they, talked about crash faults, they talked about stop fault. But also there are other faults like message corruption fault and etc., which is a part of the network communication.

But then to encapsulate a behavior that is not crash, but that is downright you know malicious they considered this notion of Byzantine generals, this story of Byzantine generals to formulate this problem. So that is what we mean by Byzantine faults or Byzantine behavior. That is malicious behavior by one of the entities or more entities that we do not know.

(Refer Slide Time: 10:34)

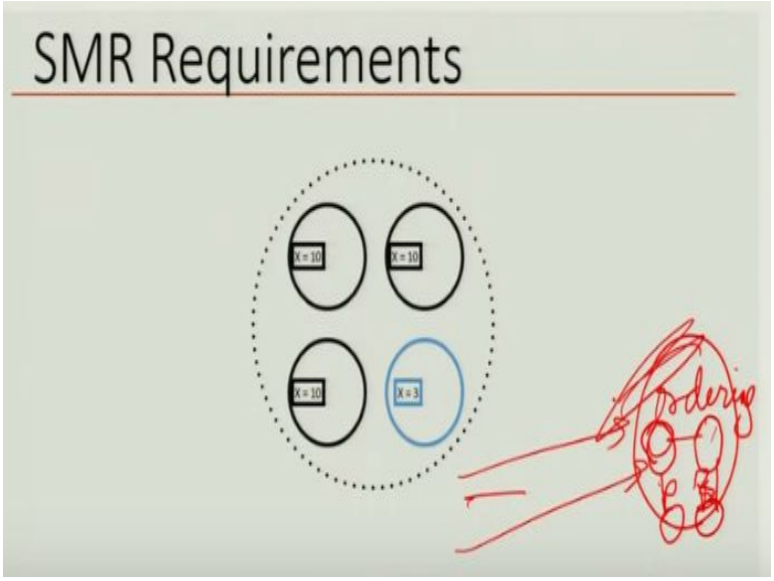


But let us say I am designing this system, then I have to make some assumption. If all my generals or all my replicas go Byzantine, then I have no way to serve the customer. So I have to question that, at most how many generals or how many replicas can go Byzantine, and I can still guarantee service correct service to the customer. And that is the problem of Byzantine consensus or Byzantine agreement problem.

So here we have a Byzantine agreement problem. Now in today's world, this was done before all this notion of malware and everything was not there. But in today's world, if a malware infects one of the replicas then it can behave

Byzantine, right. So this is a very realistic scenario in today's world that some machines may be infected by malware or somebody might do a privilege escalation and take over that machine and then not let it do the right thing. So this is quite possible with cyber-attack scenario.

(Refer Slide Time: 11:48)

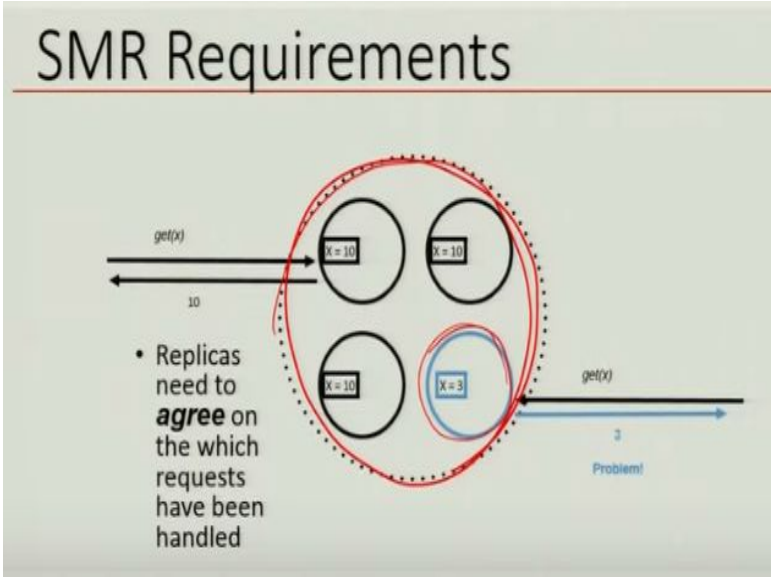


So in case of Hyperledger for example, let us say the transactions are sent to by different clients to this ordering service. So you expect that the ordinary will be done correctly, right. Now this ordering service, let us say you want to make it crash tolerant. To make it crash tolerant you need to do you need to have at least one replica right. So this is primary and backup.

So when a crash happens, you have to assume that only one we crash, both will not crash. Because if both crashes there will be no ordering, the whole thing will halt. But if you assume one crash, then you can do this. But if you assume Byzantine behavior, then as we will discuss, you will need at least four of them in order and that also by assuming only one of them will go malicious.

If you assume the two of them will go malicious, then you will need at least seven replicas. So depending on what you are going to assume and how strongly you feel about importance of the service continuation is despite some of the some of them going rogue will determine how much replica you are going to do in the ordering service. So that is the idea that we are now trying to just give illustrate to you through some examples.

(Refer Slide Time: 13:11)



You may understand it very superficially because there could be an entire class entire course on Byzantine agreement and Byzantine consensus, and so on. So it is not that you know by listening to lecture for an hour or so you will completely master the idea of Byzantine failure, but at least for this class, you should understand why you know ordering service needs to have certain level of fault tolerance.

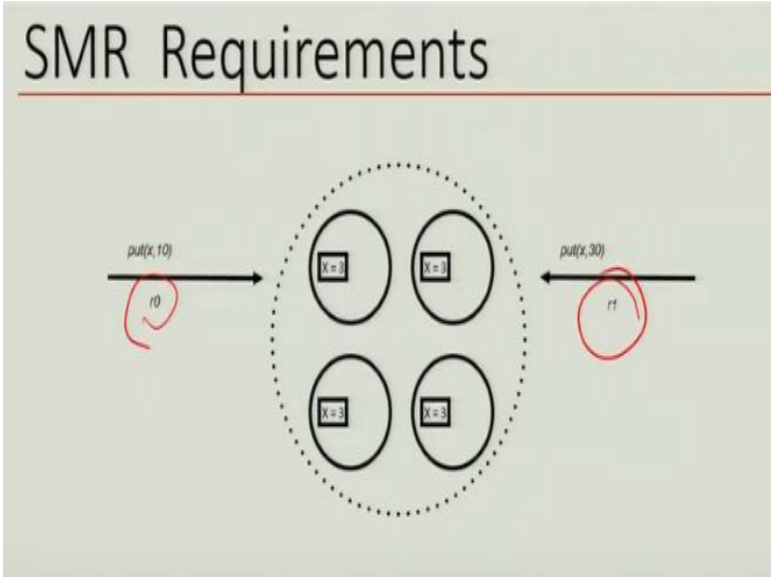
It should have crash fault tolerance. That is when if the machine that is doing ordering crashes, then at least somebody can take over that is one thing. But if you can assume that if the designer of the entire infrastructure of the blockchain besides that, there is also a possibility of malware attack or cyber-attack, then you might want to assume a Byzantine failure model. So this kind of thing you need to understand.

So here in this model, there are four replicas and then you are seeing that one of them is misbehaving. So now a client says give me value of x after the previous transaction where x value was changed. So you will get the right value. Now what is happening, one thing you need to understand is that when a client submits his request, he does not see these four things separately, right. He asks it to the ordering service.

So the ordering service or whatever service it is, will may actually first send this request to whichever server is closer to the client in terms of network sense. So maybe another client who is there who request, his request goes to this replica. Then this replica will give a wrong value, right? And that is where this Byzantine failure will fail the system.

So the point is that all the replicas need to always agree on the state and all requests must be fulfilled in the same way irrespective of who is requesting that service, when the system is in a certain state.

(Refer Slide Time: 15:39)



So now the other issue is that when you have multiple requests, how that is handled, because earlier we were talking about one is behaving improperly and therefore if the information and the request is going to that particular replica will get this different results, etc. But now we are talking about when multiple different requests are coming.

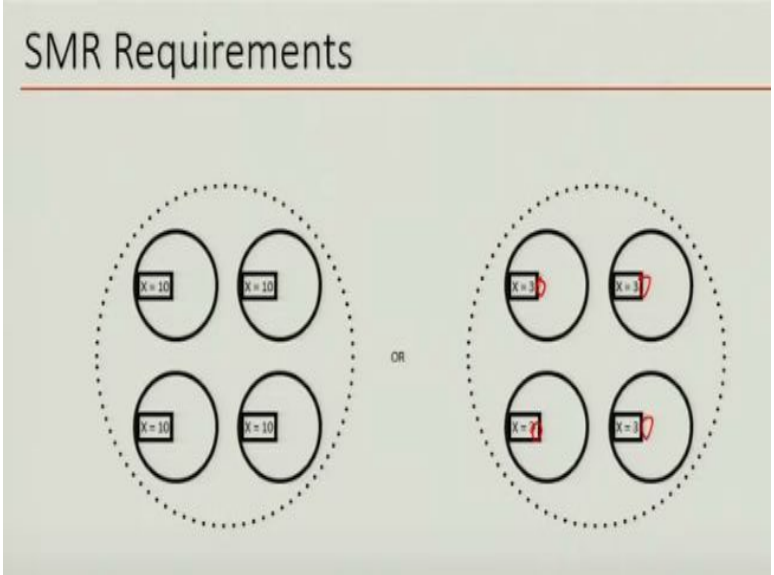
Now when multiple requests are coming on the same piece of data, multiple transactions on the same piece of data. For example, here we have one transaction from one client, who is saying make x equals 10. The other guy, another client says make x equals 30, right. Now in this case, somebody has to decide in what order these transactions should be executed, because if the order is this r 0 first and r 1 second, then x will first change to 10 and then it will change to 30.

If you if the r 1 is goes first, then x will first change to 30 and then to 10. So after these two transactions, if somebody asks the value of x, what the answer will be will depend on what would be the what is the in what order the transactions were done. What that also means that all these replicas should do the transactions in the same order. Now as I was saying that when these requests come, they may go to the one that is in closer to that client in the network sense.

So this may go to this guy on the left, and this may go to this guy on the right, right? So if this guy decides that I will do r 1 first and this guy decides I will do r 0 first, and then this guy does r 1 and this guy does r 0 later, then this guy will end up in 30. And this guy will end up in 10, which is no good, because all the replicas should have an agreement on what their state is all the time.

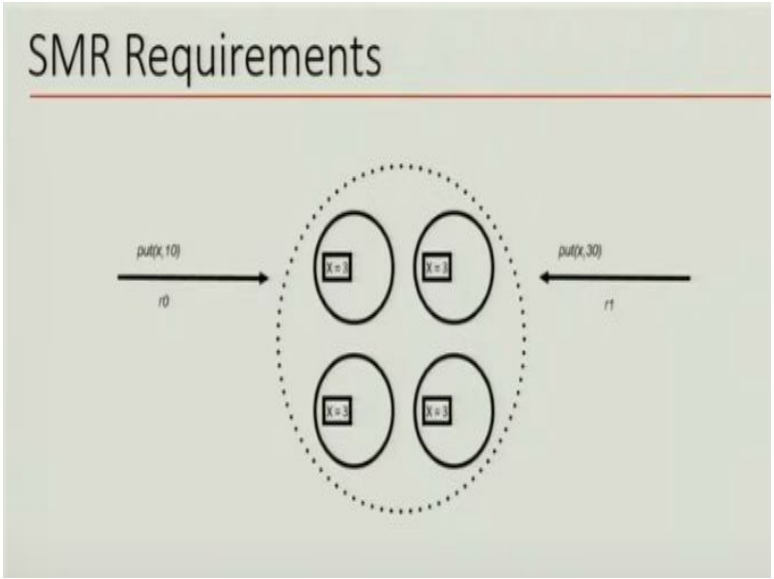
So therefore, we have to make sure that the transactions are ordered correctly.

(Refer Slide Time: 17:54)

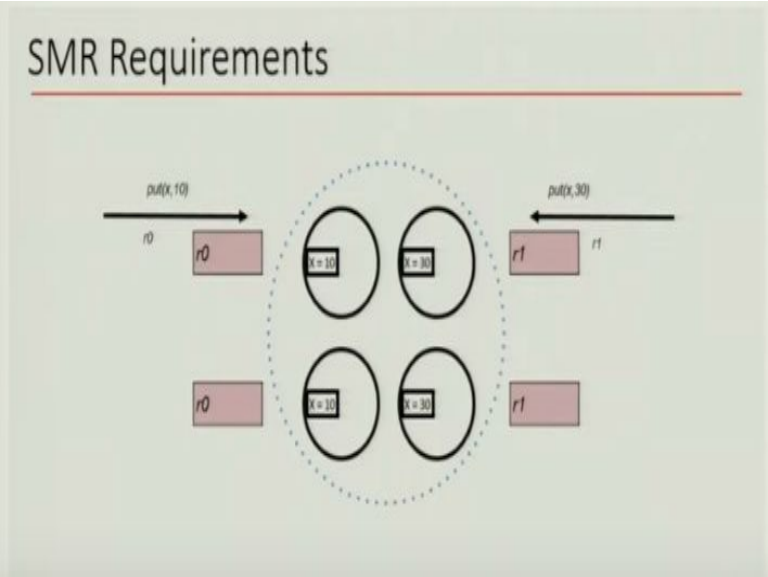


So the result would be should be one of these. Actually, this is this should be 30. But the result should be one of this, then we will say that it is correct. But if result is like half of them have 10 and half of them has 30. That is not correct. So that is what we want to guarantee that it is either after those two transactions, it is either all 10 or all 30.

(Refer Slide Time: 18:22)

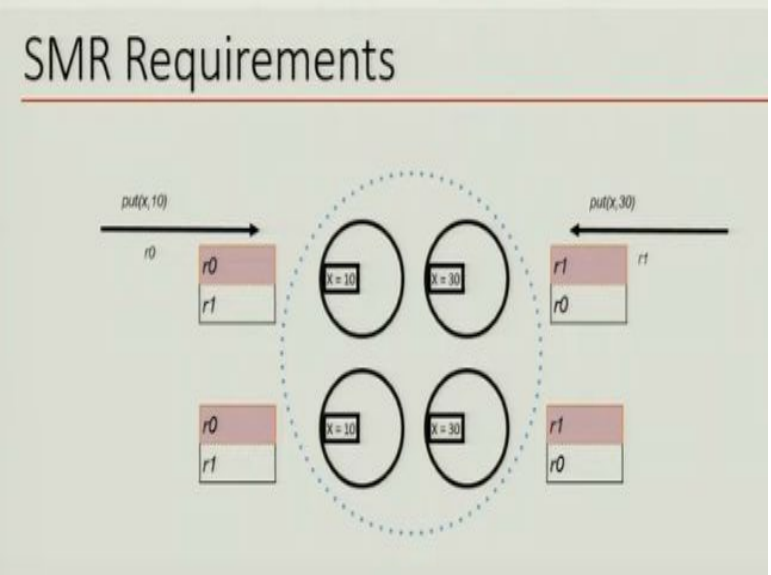


(Refer Slide Time: 18:24)



So therefore, we have to decide how to order the transactions. So there are many algorithms. So we will now see how this is done.

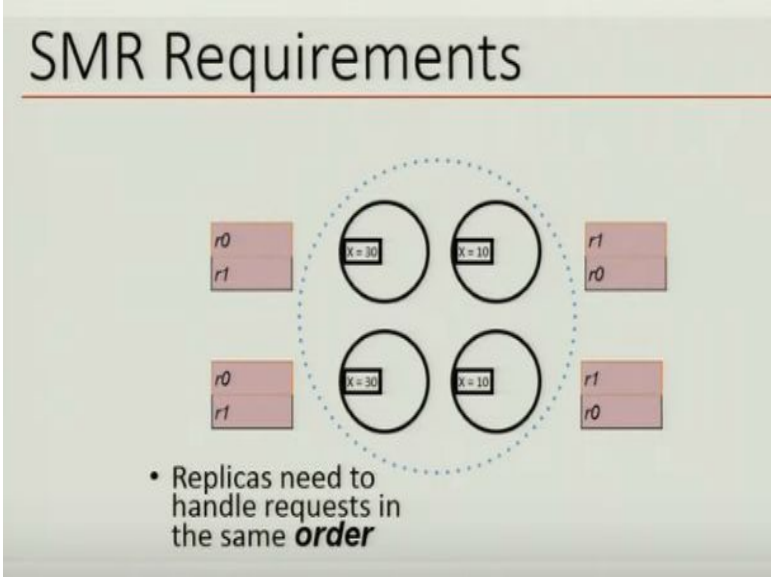
(Refer Slide Time: 18:35)



So everybody, every replica, will get the list of transactions that are under processing right now. But they got them in the different order. For example, this guy has got r 0 first, this guy has got r 1 first, so they will put them here. Now then, everybody has to decide to either do r 0 first r 1 second, or everybody has to decide to do r 1 first r 0 second.

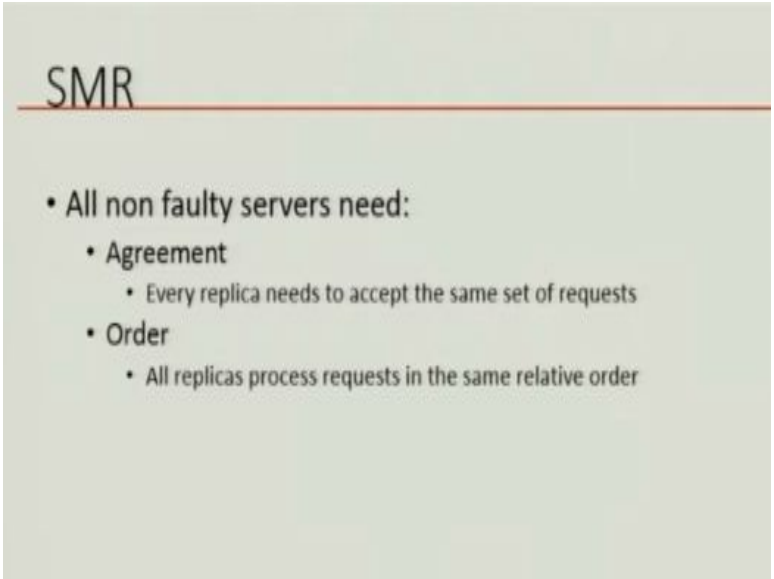
And how will they decide that because they are there, they are different servers and therefore, there has to be some communication between them before this is decided, right. So before doing the execution of the transaction, they must decide on the order of the transactions.

(Refer Slide Time: 19:22)



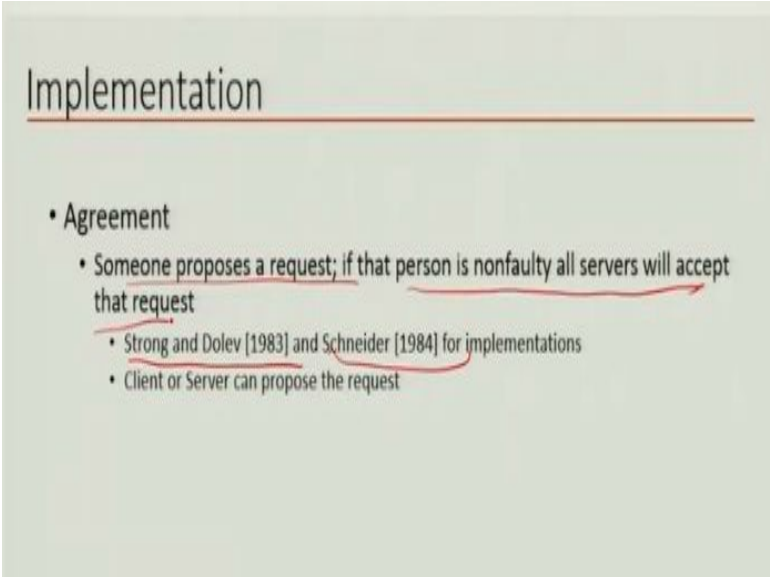
So what they will do, right they will run some kind of an algorithm.

(Refer Slide Time: 19:28)



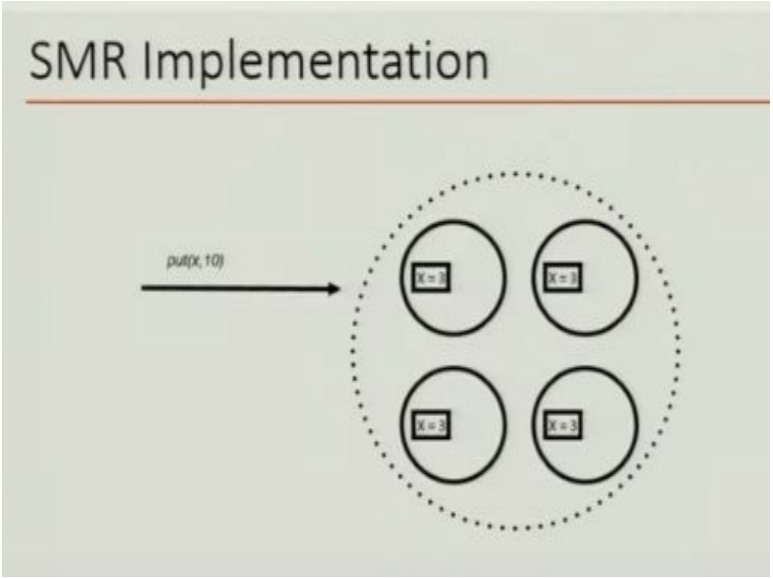
So one such algorithm before that, let us see what we want, right. So we want all non-faulty servers to agree on the order of the transactions and the and then the same set of transactions and the same relative order.

(Refer Slide Time: 19:45)

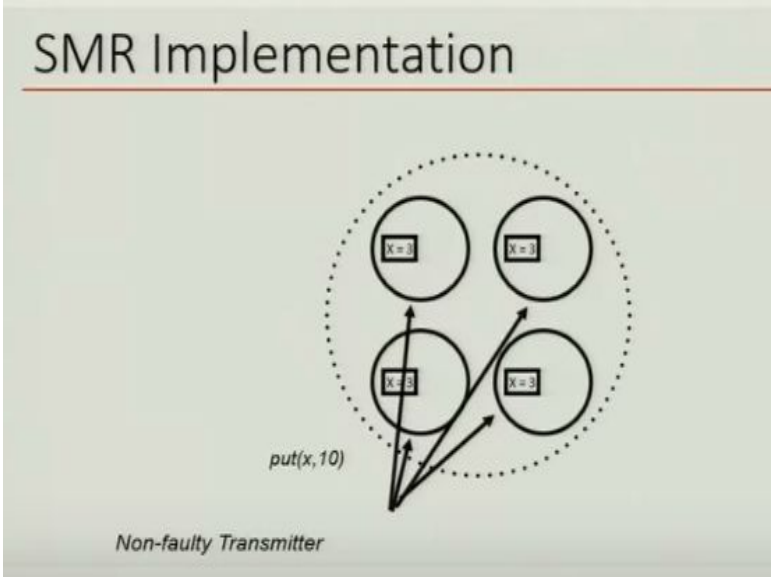


And so there is this algorithm by Strong and Dolev and then Schneider also, Fred Schneider also did some you know improvement on that. So the agreement this guarantees agreement that someone proposes a request and if that person is non faulty all servers will accept that request and any client or server can make a request.

(Refer Slide Time: 20:16)

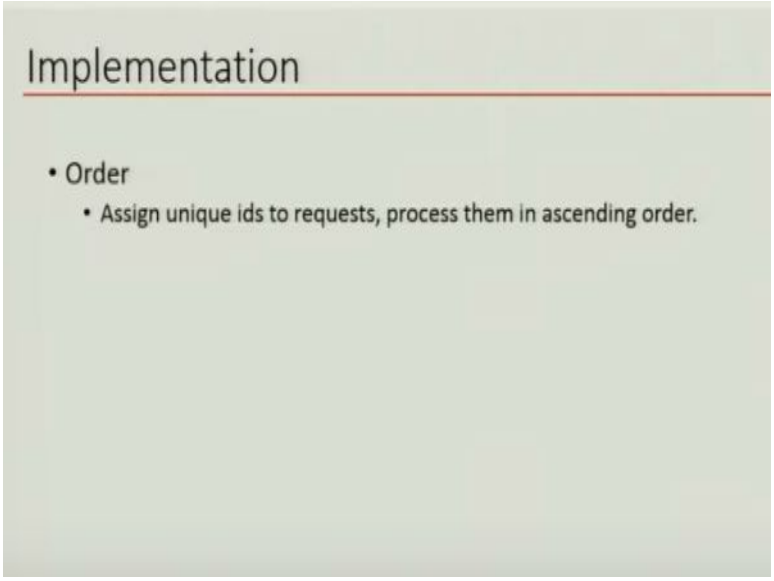


(Refer Slide Time: 20:17)



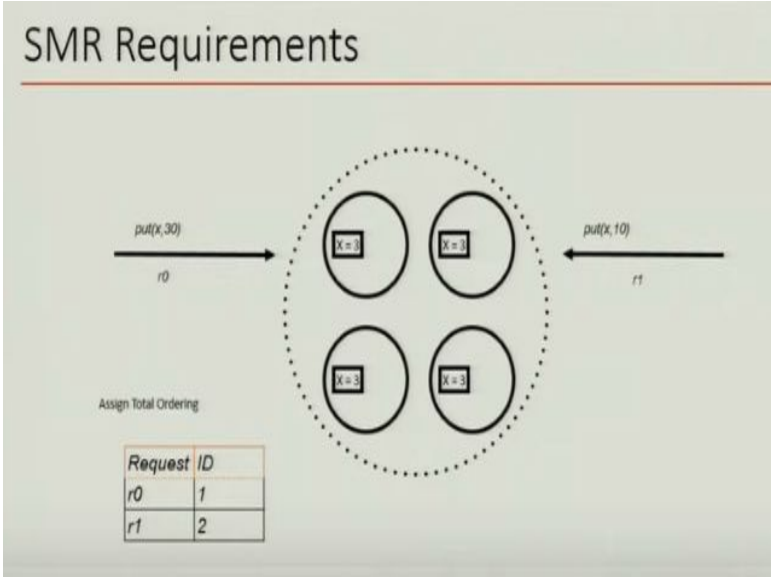
So if the non-faulty transmitter everybody will get the transaction.

(Refer Slide Time: 20:20)



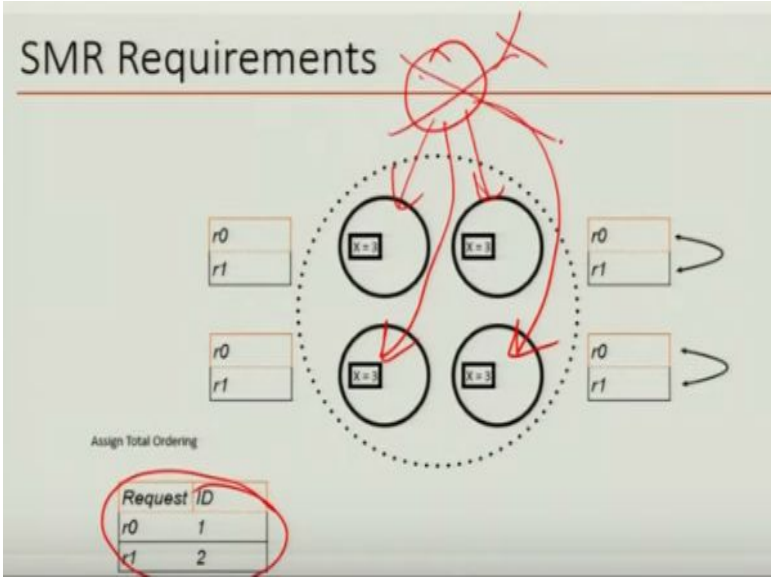
And then the ordering is done by a ID based assigning IDs. So now question is so r 0 may be given an ID and r 1 may be given an ID, but eventually everybody should decide on the same ID right. So that they, the if the IDs are given from an ordered set, then the increasing order will decide what the order of transaction will be.

(Refer Slide Time: 20:53)



So this is what we are going to do. So we want to assign total order. Now you want to assign total order. But if you leave it to each of these replicas to decide the order, then they may decide different orders. So we have to do something.

(Refer Slide Time: 21:10)

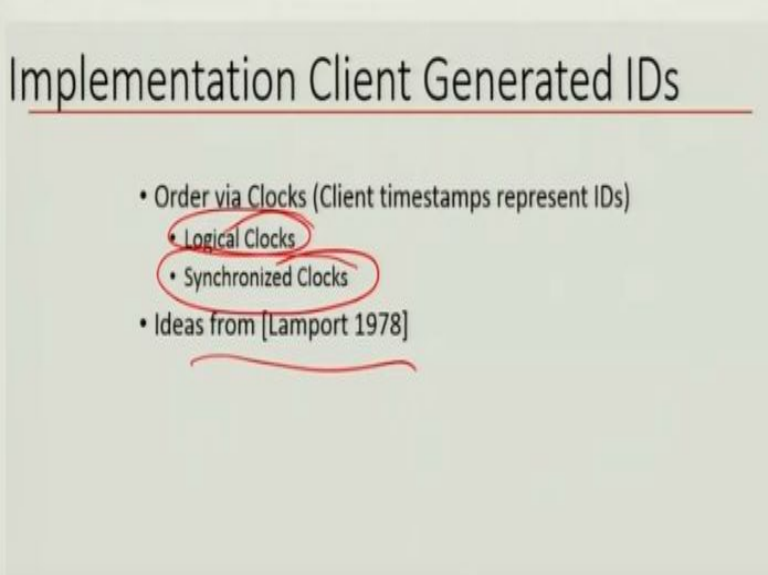


So what they will do is as follows. So they will first, so they need to reorder. If this is the final ID that we decide for everybody, then they will have to reorder the transactions. These guys are already in the order, these guys have to reorder the transaction. Now if there was a central controller here, who will tell everybody what to do, then it is no longer a distributed system.

And then and if this guy crashes, we still we are back to the same problem of not having any crash tolerance. So this cannot be the case, right. So we cannot have a scenario like this. Therefore, we must have a distributed algorithm in

which everybody decides on their own, of course by exchanging some messages, but eventually we can guarantee that everybody who is non-faulty will come to the same conclusion.

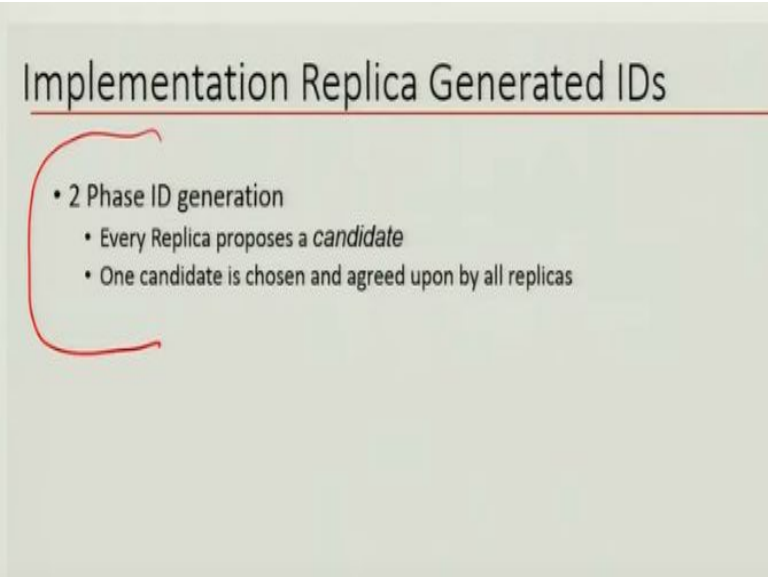
(Refer Slide Time: 22:01)



So this ordering can be done by obviously if everybody had a synchronized clock, that everybody is synchronized to the same clock then they can say okay these this transaction came at 8 pm and this transaction came at 8.01 pm. So this would be the order. But unfortunately in a distributed system clocks of all the machines are not necessarily in synchrony, right. They might have drapes and skews.

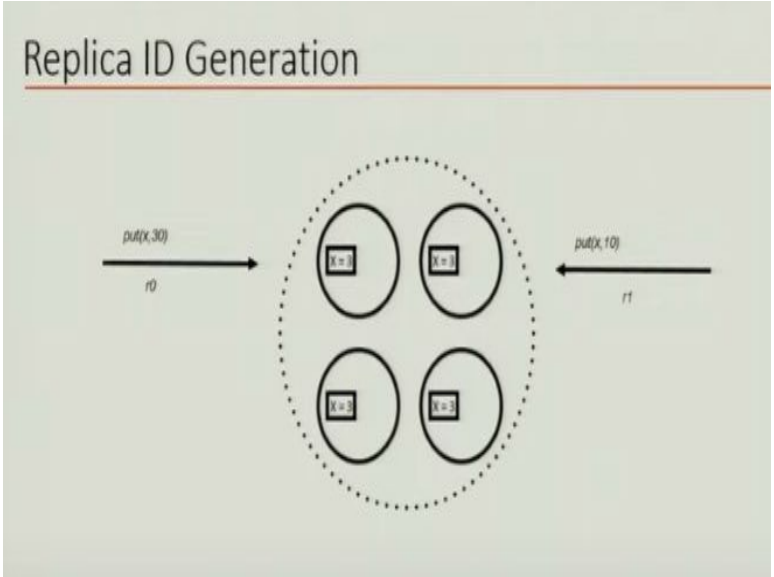
So there are algorithms for synchronizing clocks. So distributed clock synchronization is has an overhead because it requires exchanging of lots of messages. So synchronized clock is not a good solution. Logical clock is another solution. That is that was given by Lamport. But what this problem requires is not that complex.

(Refer Slide Time: 23:02)



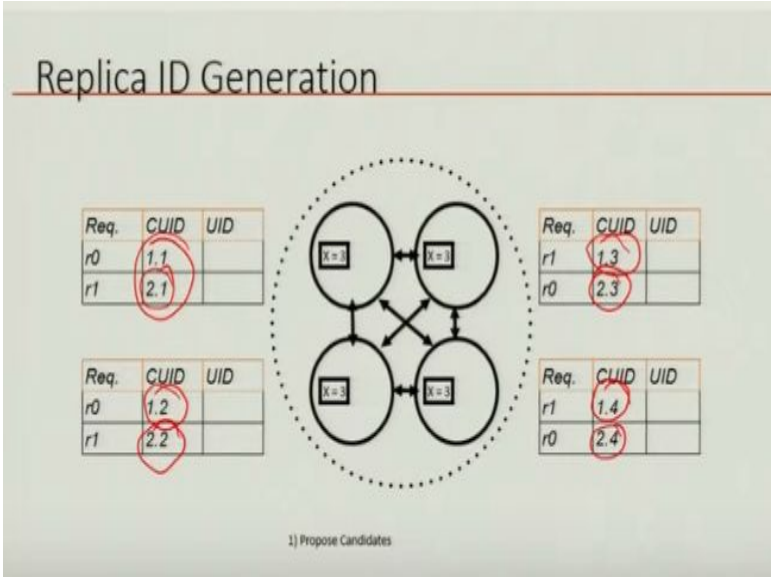
So there is a algorithm, which is called to 2 phase ID generation algorithm that was invented to solve this problem.

(Refer Slide Time: 23:18)



So what you do is that you generate first local IDs, right.

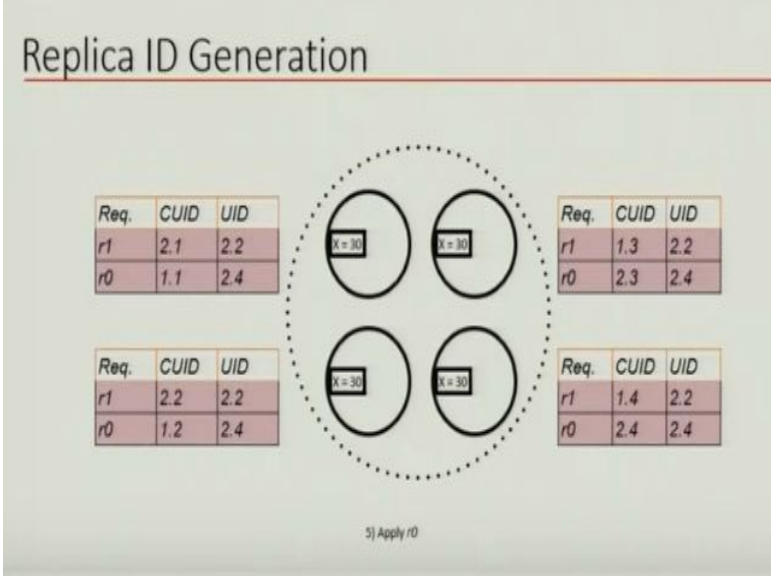
(Refer Slide Time: 23:21)



So these are called candidate IDs, candidate unique IDs. So this guy decides that the you will give r 0 1.1 and r 0 2.1. This guy decides you will give it 1.2 and 2.2. And this guy decides you will give an id 1.3, 2.3. And this will be 1.4, 2.4. This 3, 4, 1, 2 are their own IDs. So actually he is saying this is 1, this is 2. He is saying this is 1, this is 2. He is saying this is 1, this is 2, he is saying this is 1, this is 2.

Now but this is not same in all the places because for this guy, these two guys r 1 is 1 for these two guys r 0 is 1. So some algorithm has to exchange messages between them and after the messages are exchanged, so message exchange in this case will be everybody telling their local IDs for all the transactions.

(Refer Slide Time: 24:23)

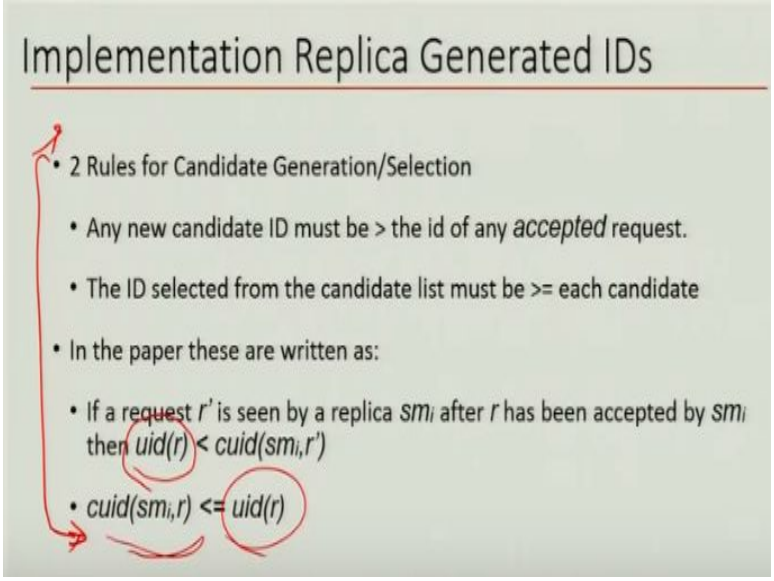


Then the algorithm says that take the highest one. So start with the one of the transaction and give the highest one. So everybody has told everybody what they have assigned to each of these transactions, right. So everybody now knows for every transaction what the id given locally, so this guy knows that r 0 has been given 1.1 by me, my this down south neighbor has given it 1.2.

On my east upwards neighbor has given me given it 2.3 and this guy has given 2.4. So then you will decide that 2.4 is the highest. So I will give 2.4 to this. This guy will also see 2.4 is the highest for r 0, he will also give 2.4. This guy already knows that this is the highest because all the messages are have gone to him. So he will also give 2.4 and this guy will already has given 2.4. Then they will consider r 1.

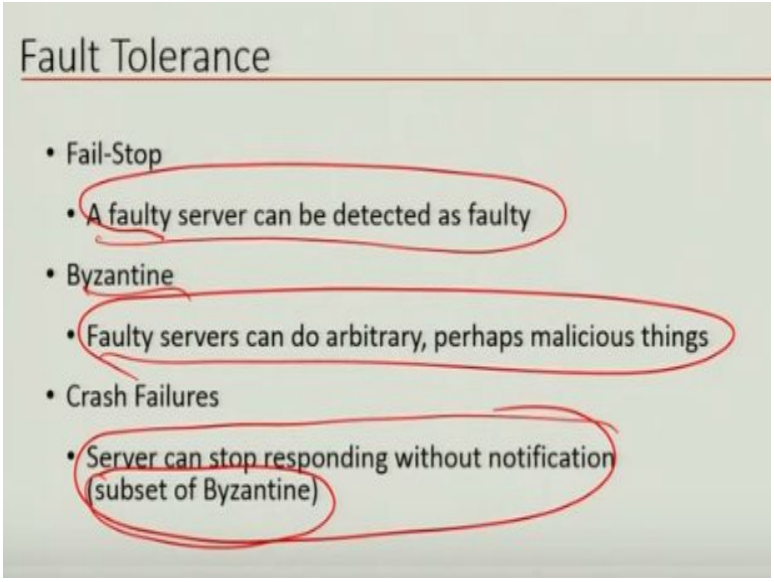
So this guy will say r 1 has 2.1, 2.2, 1.3 and 1.4. So 2.2 is the highest. So I will give r 1 2.2. And everybody will do the same thing. So at this point 2.2 is smaller than 2.4. So therefore, the ordering will now become r 1 first r 0 second. So that is how, so this guy will now reorder the transactions, and then r 1 will be executed first, which will make x equals 10. And then r 0 will be executed and which will make x equals 30. And this will be done at everybody because everybody knows the same ID, global ID for each of the transactions. So ordering problem is solved this way.

(Refer Slide Time: 26:23)



So this is the more formal discussion formal way of expressing what we just described. So you do not need to really worry about this. This basically says that, you know when you give a global ID of a transaction, then you must make sure that it is the less than the candidate ID for that. And then every candidate ID is less than is equal to the UID r so UID r would be an upper bound on all the candidate IDs for the same transaction, same request, etc.

(Refer Slide Time: 27:04)



So now this system, when we describe this system, how this happened, we did not assume any fault. So we assume that all of them are working fine and there has not been any crash or fail-stop or Byzantine behavior right. So fail-stop means a faulty server can be detected as faulty. So it may actually declare that I am going down So everybody knows that it has gone down.

Crash failures on the other hand will be when the server stop responding. So it may be failed or it may be actually behaving, it may actually be highly alive, but it is saying it is not responding intentionally. That is sort of a Byzantine

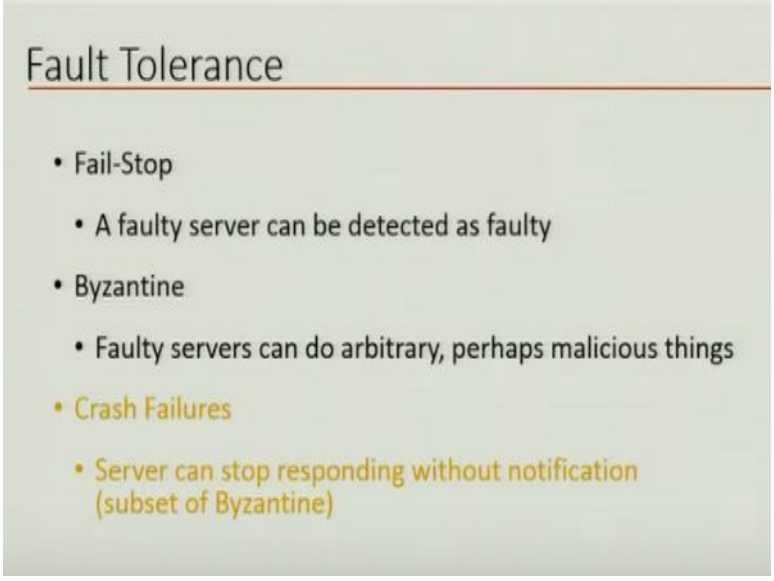
behavior. And the Byzantine behavior which will be more general that is they can do arbitrary behavior. They can send you wrong information.

In the earlier this picture that we were showing there was no wrong behavior. Everybody was honestly sending their CUIDs to everybody and that is how it was simple.

(Refer Slide Time: 28:08)

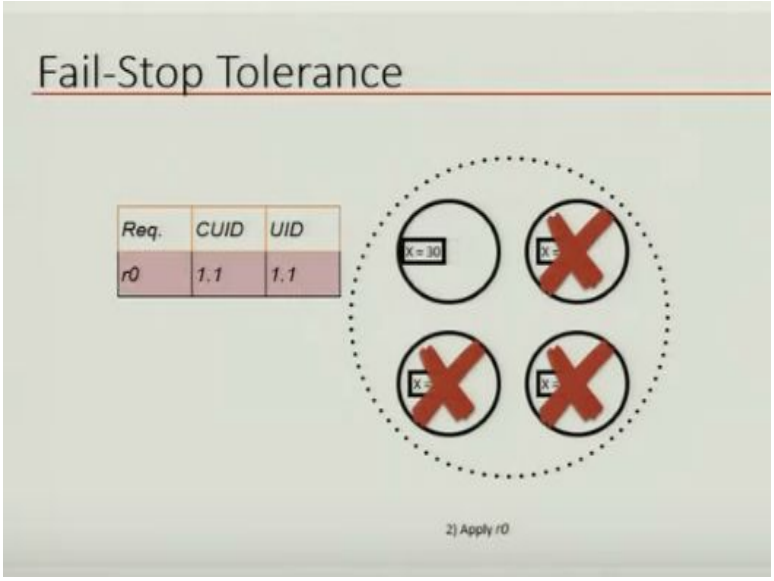
So here, let us say we have a fail-stop, right?

(Refer Slide Time: 28:13)



So fail-stop is when you can detect something is faulty. So you do not wait for yeah, we do not wait for any message from that.

(Refer Slide Time: 28:22)

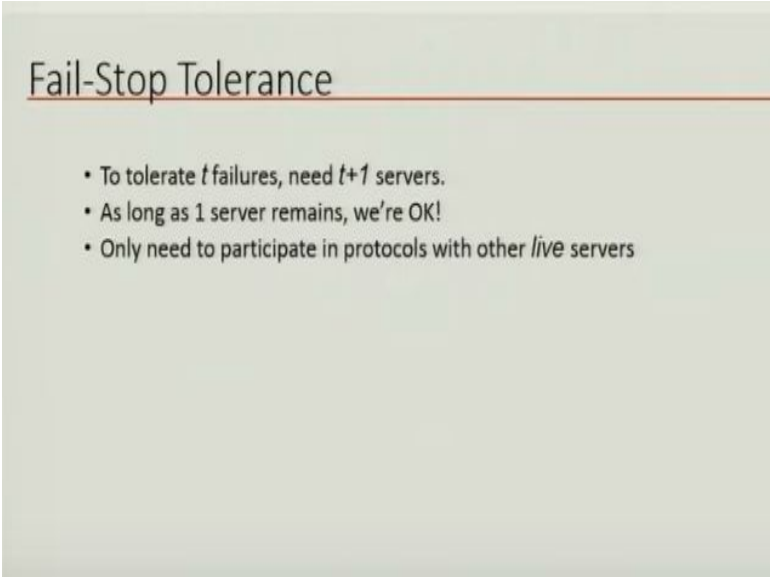


So in fail-stop tolerance, so let us say three of them fails, and you have this transaction. And then this guy is the only one working so you accept r 0 and that is it. And there is no problem.

(Refer Slide Time: 28:44)

So now all the other ones will have the same old value of 3. So now if this guy also crashes, then game is over, right? So fail-stop tolerance will only work at least one replica should survive you know to be self-stop tolerance.

(Refer Slide Time: 29:07)



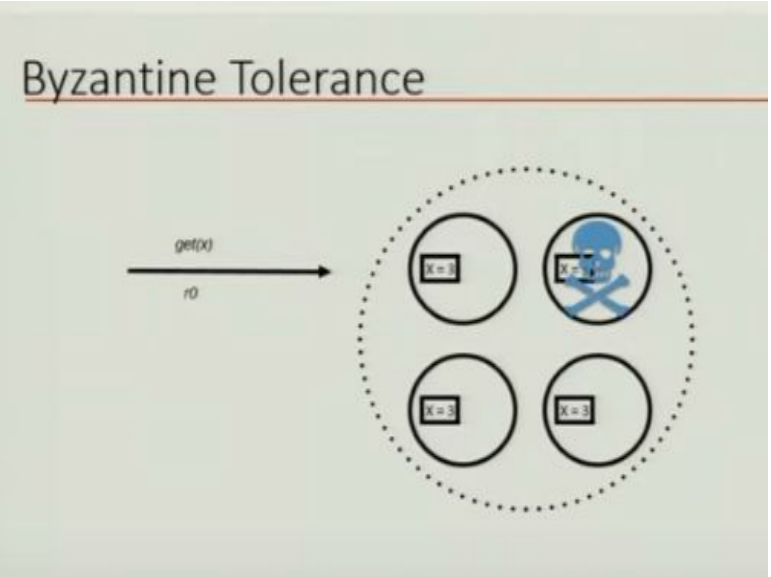
So to tolerate t failures we need t + 1 servers in fail-stop tolerance. As long as one server remains, we are okay. Only need to participate in protocols with other life servers. It is the easiest to implement. You just have to have extra. If you assume t of them can fail then t + 1 servers is enough. So when we come back, we will talk about Byzantine tolerance for the same setup.

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

Lecture - 24

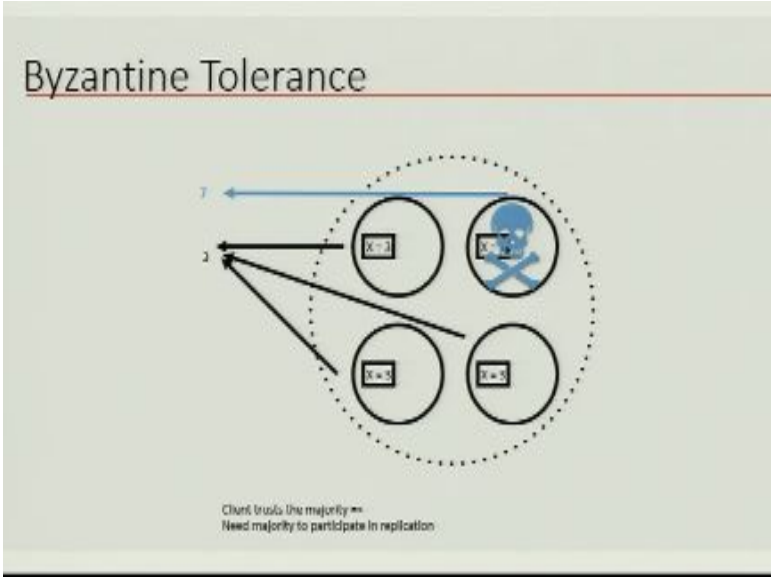
Welcome back. So we were talking about replicated state machine. And we talked about how to order transactions on them. And now we also talked about fail, fail-stop failure. And there we saw that t for, if you assume t of them will fail, then t + 1 servers are needed to tolerate that. Now let us talk about Byzantine tolerance.

(Refer Slide Time: 00:40)



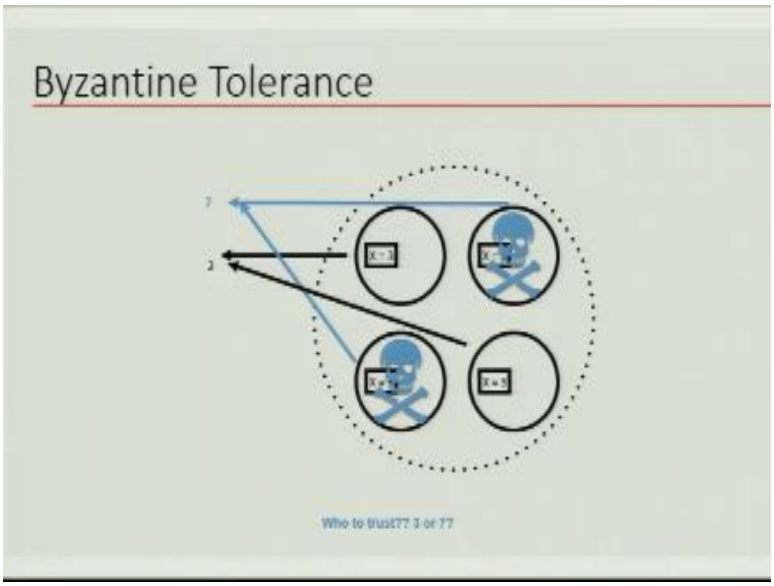
So again, we have four copies of the same state machine. And now let us say one of them decides to behave arbitrarily. So it is a Byzantine node. So now a transaction comes.

(Refer Slide Time: 00:56)



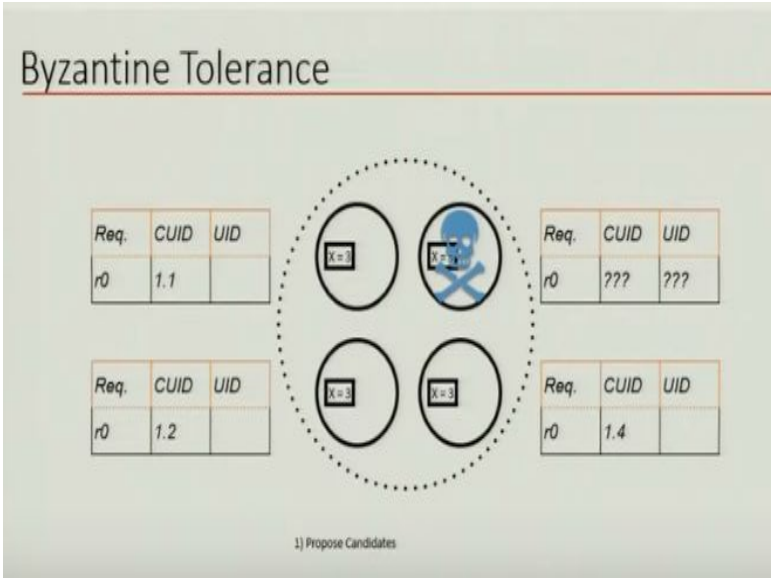
Now all the three nodes will give you the correct value that are not affected by Byzantine, but the one that is Byzantine may not answer you or may answer you wrong. So how do you decide? So client will have to do a majority based decision. So client will say, whoever is whatever is the majority saying I will go with that. So in this case, the correct value 3 will be decided by the client as the correct value correct answer to its query.

(Refer Slide Time: 01:29)



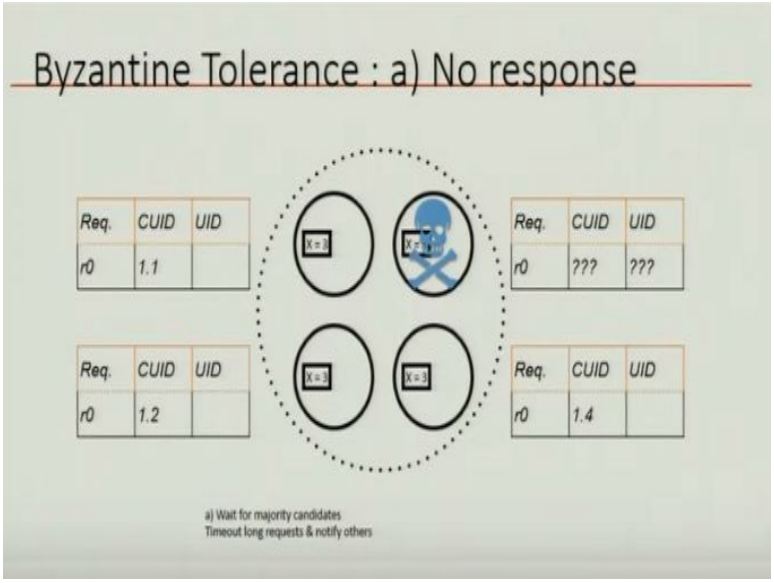
Now if two of them does fail, as Byzantine, and two of this Byzantine guys are under undergoing the effect of the same malware, so they decide to actually fool the client with the same value. Now we have a problem because majority voting will not work anymore, because there are two 7 answers and two 3 answers and the client will not be able to make a decision.

(Refer Slide Time: 01:59)



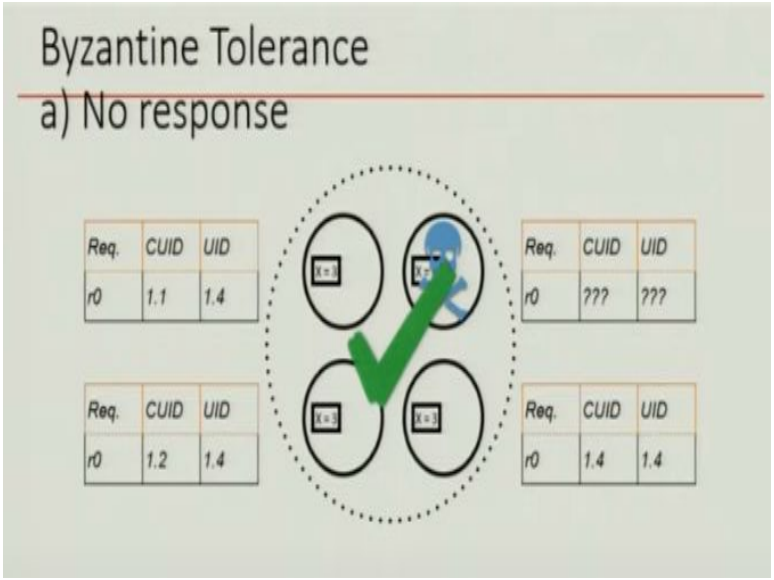
So now again so let us say we want to try this algorithm as before. Now I have r 0, and r 0 has been given this local IDs. But the Byzantine node may or may not give it any ID, or give wrong ID. So in this case, let us assume that the Byzantine node decides to keep quiet and not give any response.

(Refer Slide Time: 02:17)



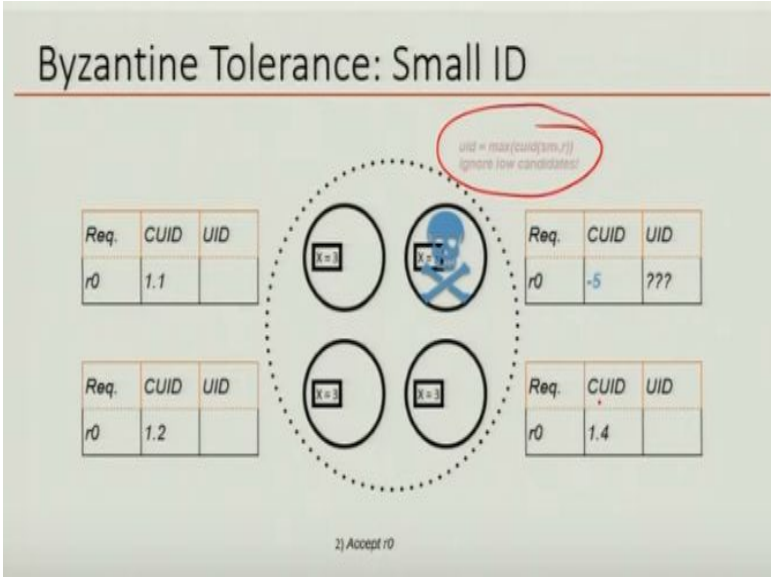
So then you basically get the majority of the candidates, and then you use timeout to decide that the fourth answer is not coming. And then you decide the majority is this. So I will based on that I will give it a UID.

(Refer Slide Time: 02:45)



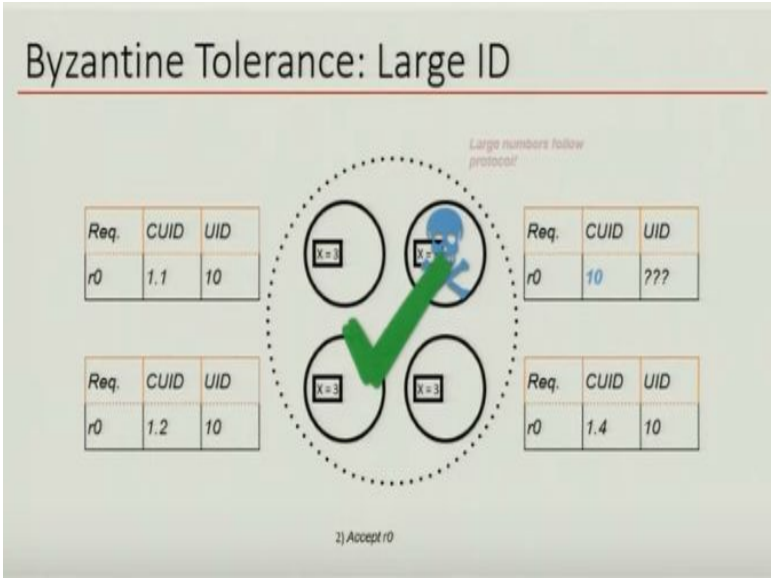
So in this case, 1.4 was the highest so you give 1.4, it worked. So one node being Byzantine and not giving any response did not affect your ability to give ID to a agreed upon ID to the transaction.

(Refer Slide Time: 03:07)



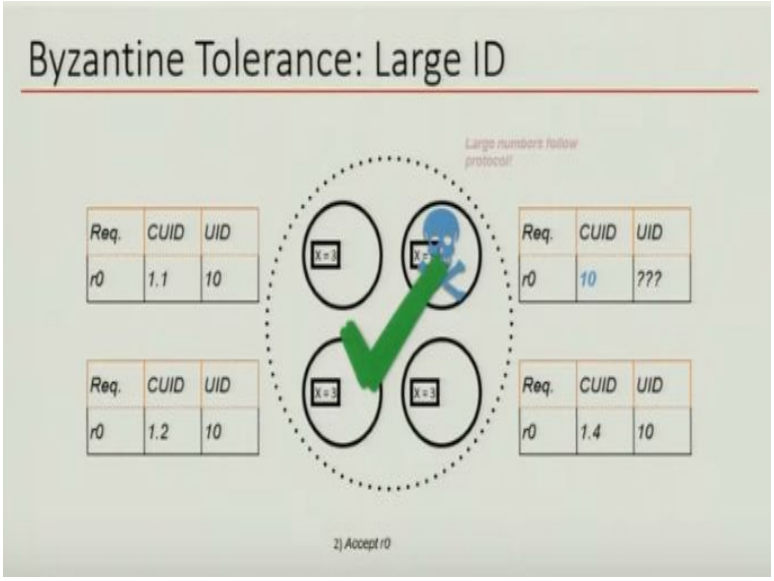
Now suppose the this other guy, the Byzantine guy wants to behave more maliciously. So he wants to confuse you. So he gives you an ID but he chooses a small ID because he does not know what ID others are choosing. So he gives you an ID let us say -5. Same issue that the algorithm says that you have to take the UID as the maximum among the all the local IDs. So even if it says -5 no problem 1.4 is still the highest.

(Refer Slide Time: 03:40)



So I am going to choose 1.4 no problem, right. So this also worked.

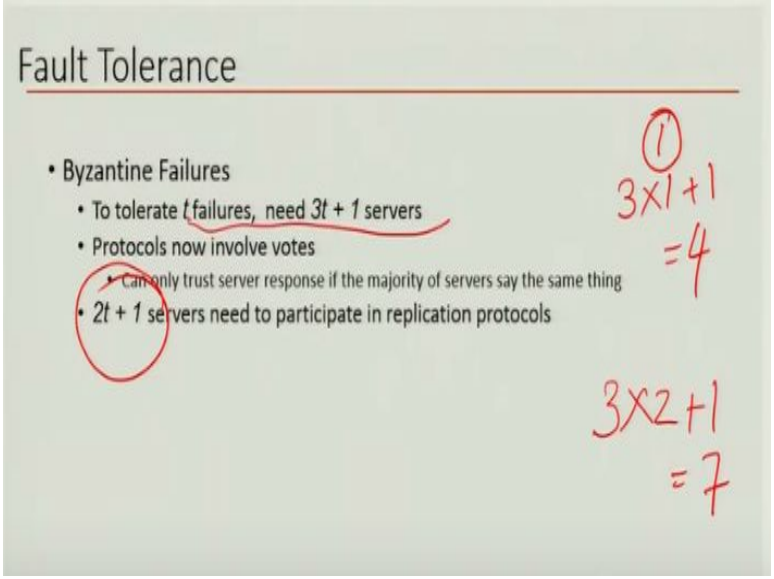
(Refer Slide Time: 03:44)



What if this guy gives you large ID, right? So he gives 10. Now based on the algorithm, now you have to decide on the ID of 10 for everybody else. Now this was fine as long as there is only one transaction. Otherwise, if this happens again for r 1, again the this guy can actually give you a large ID and make r 1 have a different ID. But that does not make any difference because all the other guys will decide the same ID for each of the transactions.

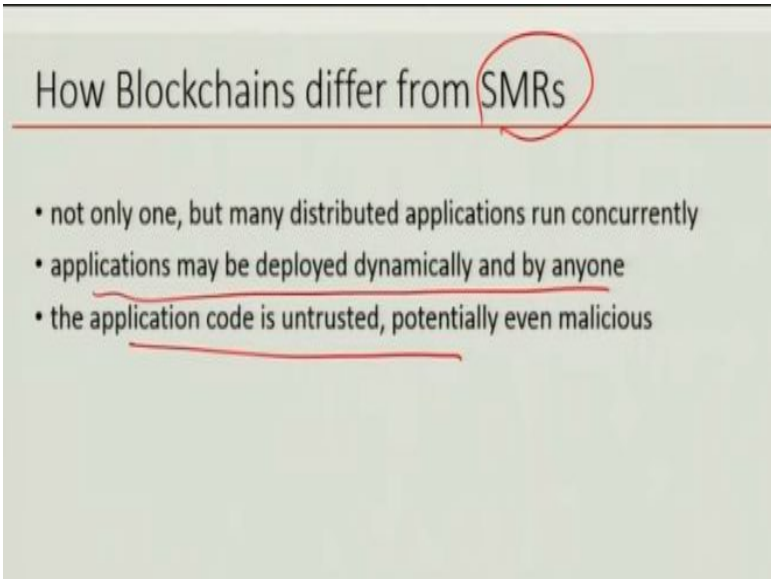
Now that ID might be dictated by the Byzantine node because he is choosing very large ID, or it may be one of the IDs that these three guys give. So that will not be a problem.

(Refer Slide Time: 04:36)



So to tolerate t failures in this case, you will need 3t + 1. So we assume that one node goes Byzantine. So we need three times 1 plus 1 that is 4 servers. So that is what we saw there. But when you saw two failures, we saw that 3 times 2 plus 1 is 7, but we had only 4. That is why we were not able to proceed. And so 2t + 1 servers need to participate in the replication protocol.

(Refer Slide Time: 05:11)



So now you understand a little bit about the state machine replication and the models for different kinds of faults. So in case of Hyperledger, this applies only to the ordering nodes. The ordering nodes are involved in making ordering decisions and they have to tolerate Byzantine failure provided the designer of the blockchain decides that there is a possibility that this ordering nodes are not necessarily extremely highly secured.

There may be patches that are missing and therefore they may be infected by malware or it may be taken over by some rogue entity and therefore they are not going to behave properly. Another possibility is that this ordering nodes belong to different organization. For example, if this blockchain is spanning three banks, then each bank may actually provide one of the nodes in the ordering service.

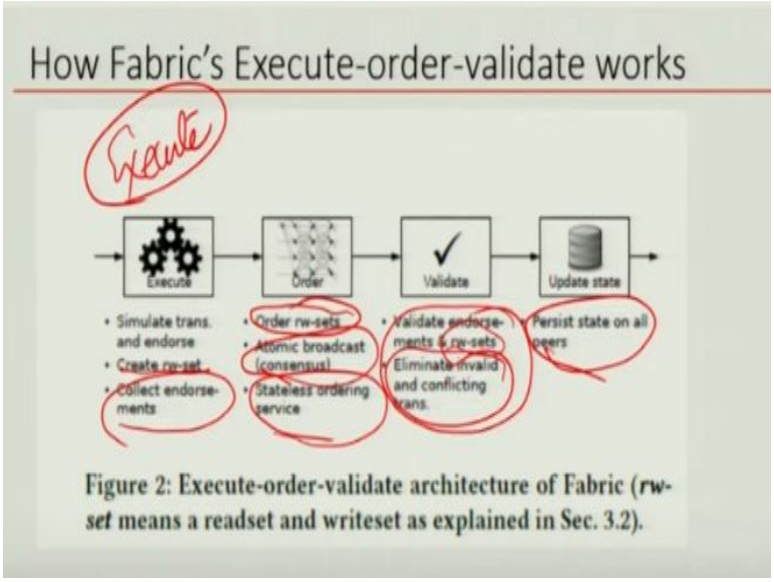
Ordering services is a replicated servers, set of servers, but each of them are coming from different organization. And you are not trusting the other organization. Maybe they will not do anything because of their business ethics or whatever. But you may not you may agree among each other at the business level, that let us not trust each other, let us have Byzantine tolerance.

But if I am going to assume that two of you will behave rogue, then I will have maybe 7 servers to do ordering service. If you if I assume one of them may be going rogue, then I will have 4 etc. They can make that decision. Now blockchains is more than just you know simple replicated state machines. Because replicated machines are just one service like a database service or something running in each of them.

It is the servers associated with the database, let us say. But in blockchain there may be many different services or applications running. As I was saying that same ordering service may be used for one banking, interbanking transaction, log keeping. Another one for supply chain and etc. All of them in the same infrastructure. So the ordering service may be associated with all of them.

Also applications may come and go as applications may be deployed dynamically. And then the application code is untrusted and potentially even malicious, which in this case in Hyperledger it does not matter because the ordering service does not run any of the application code. But that is the reason why ordering services separate.

(Refer Slide Time: 07:54)



So coming back to a picture similar to what you saw before. In Fabric, Hyperledger Fabric, we have execute. That is where the endorsers execute or simulate the execution of the transactions and create the read/write sets. So they create the read/write set and send the endorsements.

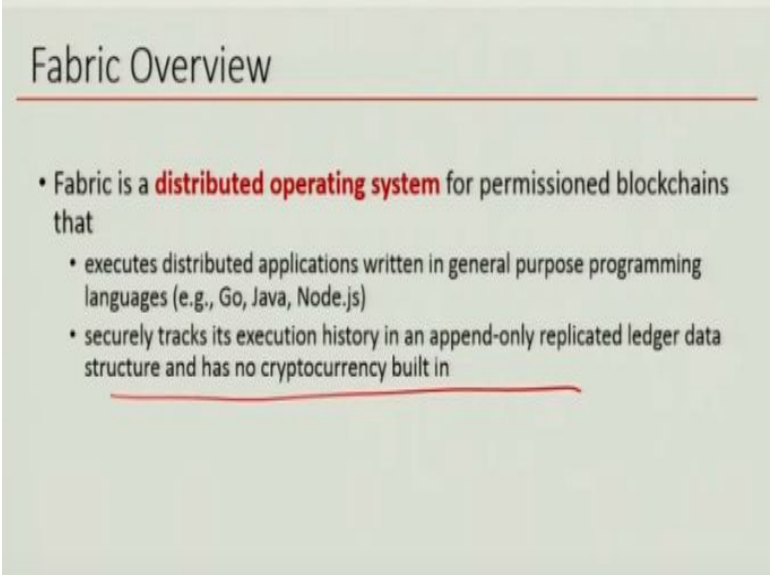
The ordering service then only order based on the read/write sets and do an atomic broadcast among each other and do a you know decide and then this ordering service is stateless in the sense that it does not keep the state at all. It is totally agnostic of what application is running, what data is being kept, etc. It only does ordering.

And then the nodes that are validators, they will validate that a transaction has all the necessary endorsements, and then the read/write sets have the right versions of each of the piece of each piece of data. And then it eliminates invalid and conflicting transactions. And then after that, they send it to every peer.

And then every peer will persist the data by adding to the blockchain the block that was given by the ordering service. And with a bit vector that has one's and zeros. One means that the first if it is 1,0,0, that means first transaction is valid second, and third transaction is not valid, etc. So that information goes to every node.

And every node keeps the block with the bit vector so that later on when you retrieve the blocks and want to see which transactions were valid and actually was persisted, and which transaction was tried, but it did not persist all that information there. And this is very important for auditing, especially security auditing. So that is the basic idea of the execute-order-validate framework.

(Refer Slide Time: 10:00)



So Fabric, the Hyperledger authors, they actually consider Fabric more like an operating system which runs applications. So why they are calling it an operating system like why is it not Ethereum or bitcoin not considering it as an operating system? In operating system you have many applications running. Each application may be consisting of one process or multiple processes.

And these applications as they run the processes run their own address space. So there is full confidentiality between two processes. So what is happening inside this process, what is in its memory etc., you know unless there is a cyber-attack based exfiltration of some kind, buffer overflow, etc., these processes are silos in which the application runs.

Or if an application is multiple processes, they may exchange data with each other through shared memory or through, you know RPC etc. But all the other applications do not know what is happening there. In Ethereum, this

ability to silo multiple different applications that are running on the Ethereum is not done. So every smart contract for every application are on the blockchain, every node sees them.

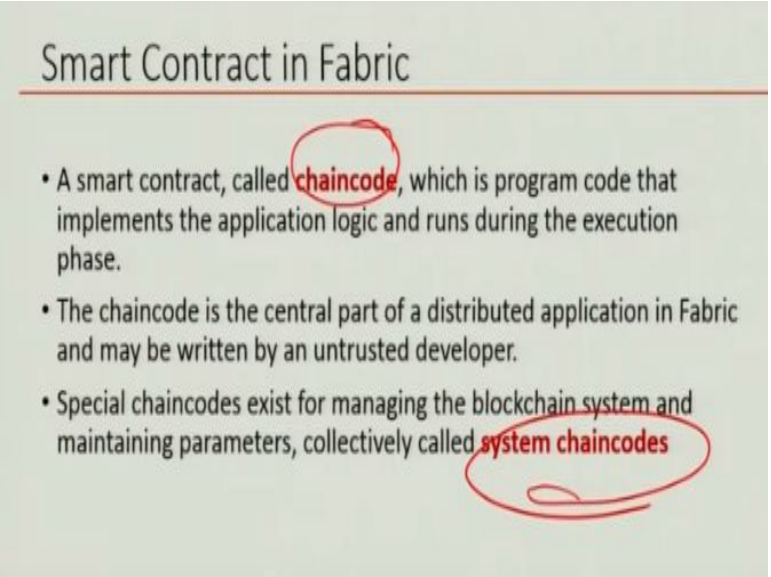
So therefore, we cannot compare Ethereum as sort of like a operating system where if I create a Hyperledger infrastructure, I can have multiple channels, running multiple different applications and the chain code associated with each application will be confidential to each other. So therefore, it is more like an operating system which is enabling multiple applications with full silos.

And they are reusing the infrastructure such as the identity service or the service that gives digital certificates. The ordering service which may be common between all the applications and the actual infrastructure may be all common the network and the actual servers. But each server is running this applications, the chain code in a in a way that you know the chain code runs in Dockers.

So they are siloed out from each other. And also there is a common code that is called the system chain codes that basically does for example, things like validation, etc. These are you know put in as libraries. So it is very much like an operating system. Also in another sense, it is an operating system. Operating system can run applications written in any language. And here also the same thing that they have chosen by design.

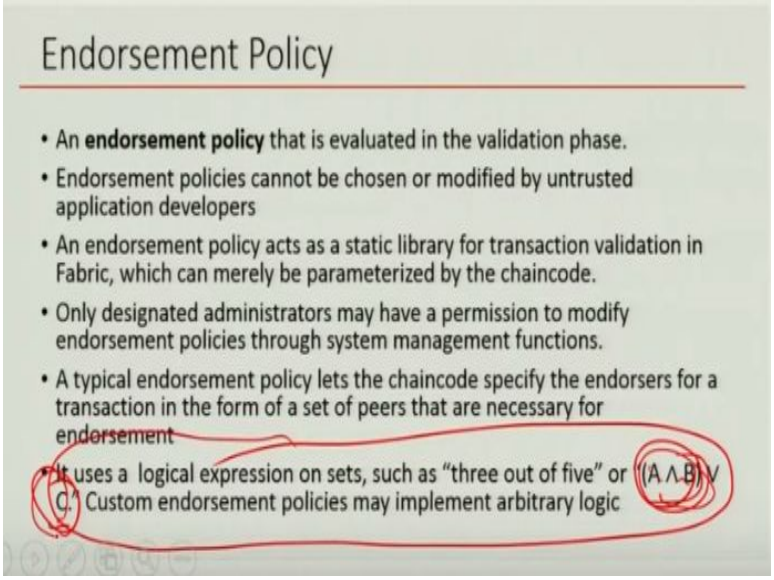
And then you can have the additional thing that execution history is kept in replicated ledger which is tamper resistant and so on. And it has no cryptocurrency built in.

(Refer Slide Time: 13:09)



So again the smart contract is called chain code which basically implements the application logic and runs during execution phase. The chain code is the main thing and fabric and there are also system chain codes as I said that basically are using managing the blockchain system, maintaining parameters etc., and they are called the system chain code.

(Refer Slide Time: 13:36)



So endorsement policy is something that we talked a lot about. But we never explained how this endorsement policy is set, who sets the endorsement policy, what exactly is the effect of an endorsement policy etc. So endorsement policy is evaluated at the validation phase which is a pre commit phase. And the endorsement policy cannot be decided by chain code developers.

So let us say you are a bank and you are running a banking application on Hyperledger. So and the code that is running in the application are designed by vendors, right? So you have you hired some coders, they write the chain code in Go or C or C++ whatever. And they will put them in Dockers, etc. But every time you want, let us say a transaction to be to be endorsed, who will endorse the transaction?

Whether it is just one node or whether it is three nodes and which specific three nodes etc., is part of the endorsement policy. And that is a that is based on the business logic and business policy of the application and the business that is running that application. So they might say that for any transaction below, you know which has an associated value of say 10,000, it has to be endorsed by two nodes and to this kind of specific nodes.

If it is above 10,000 to 100,000 it has to be endorsed by at least three nodes. It is like check endorsement in a business. And that decision is encoded in the application. So endorsement policy becomes part of the application, but

it is not part of the chain code that does endorsement. So when a client, let us say wants to do a monetary transaction, let us say like Internet Banking.

He says that from my account send 1 lakh to this other account. Then maybe the policy is that it has to be endorsed by a manager node and a sub manager node and maybe a vice president node or something. So any nodes that are that is running a chain code on behalf of the manager, sub manager etc., will actually be getting that, so client would the client the program that is running on behalf of the client will know that for this kind of transaction I need these three endorsements.

So it will send this transaction in for request or proposal to these three nodes. These three nodes the chain code in there will pre execute the transaction to see whether the balance is available on the user's account and whether the and what would be the value after the transfer and what would be the value at the other place after the transfer and then it will create the read/write sets and then it will digitally sign it.

And then the client gets endorsement back. This endorsement will come from all the three nodes that are supposed to be part of this endorsement policy. Only then the client will know that, well by client I do not mean the human sitting at the terminal, you know logged into this application, let us say through a web front end.

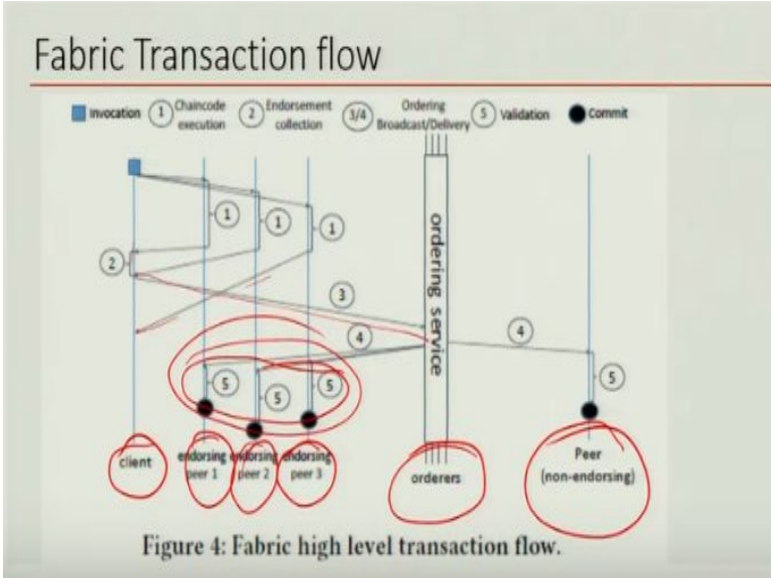
But the application that is running on behalf of the client would know that now I have gotten my endorsement policy has been satisfied. However, then the client will send it to the orderers and orderers will order etc. But as it goes to the validator nodes, the validators will also check whether the endorsement policy has been satisfied for example for this kind of transaction.

So the validators know the endorsement policy for each type of transaction. So they will check whether that has been met. Then they will check whether the digital signatures associated with each of the endorsements check out. And then they will check the read/write sets have the right versions etc. So that is what endorsement is all about.

And normally this endorsement policy is expressed by what is called monotone logic. So like three out of five, or it has to be either these two nodes, or this node may be a very high privilege node. So if this guy endorses, then we do not need these two nodes to do anything etc. So this may be two sub managers, and this may be the manager. See, if two sub manager nodes do it, then it is fine.

But if it is one sub manager, that is not enough. But if the manager himself does it or the program running on behalf of the manager does it then that is fine.

(Refer Slide Time: 18:53)



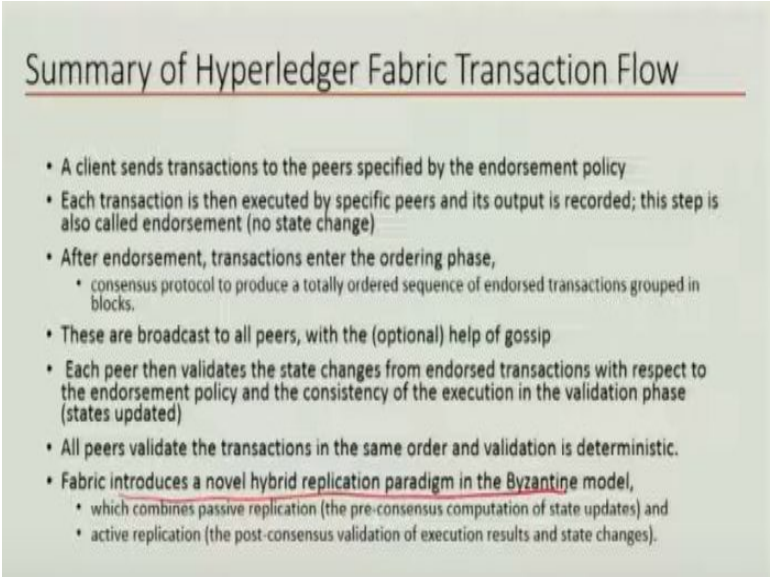
So the transaction flow here looks like this, that you have the client. So this is the client. It invokes the transaction. And then it sends the transaction to the endorsing peer 1, endorsing peer 2, endorsing peer 3. Endorsing peers will, as I said pre execute etc. and then send back endorsement. All this endorsement sent to sent back to the client.

At that point the client will all the three endorsements who come or maybe the endorsement policy is just two. So as soon as it gets two of them then it will send it to the ordering service. The ordering service is maybe made up of multiple nodes because of all that fault tolerant issues that we talked about. And then ordering service will order them and then it will send it. So it will then send it to the validator nodes.

So validator nodes are could be the same nodes as endorsers, but different chain codes So therefore, you know they are running in different Dockers. So same servers are being used for multiple purposes, but they are in silos so they do not talk to each other, see each other. So therefore that is fine, or it may be different servers, it does not matter. But in this case, let us say this is the same nodes as validators.

So then validators will send this to commit and then the and this commit will go to all the other nodes, right. So and then all the stuff will be done.

(Refer Slide Time: 20:37)



So here, I want to summarize the Hyperledger Fabric transaction flow. So a client sends transaction to peers specified by the endorsement policy. Each transaction is executed by specific peers, and its output is recorded. But nobody changes anything in the database or in the blockchain. So we call it a simulation of the execution.

And this step is called the endorsement and there will be no change of state in the terms of the state of the blockchain or any of the data in the database. After the endorsement, the transactions will enter the ordering phase and the consensus protocol will produce a totally ordered sequence of endorsed transactions grouped into blocks. Now these are broadcast to all the peers with the help of a gossip protocol.

So gossip is a protocol by which the node who wants to spread the news to everybody, he will tell its neighbors. And the neighbors will then tell its neighbors and neighbor will not tell back to the neighbor that told it, right. So this is how the gossip protocol works. So they broadcast to all the peers. Each peers then validates the state change from endorsed transaction with respect to the endorsement policy.

And then the consistency of the execution of the transactions by looking at the version numbers of the states of the data that has been that is going to be is being read or written due to this transaction. The ones that are not correct, they will be marked as invalid transaction. And then all state changes for the valid transactions will then be made. And that is what we call the states are being updated.

And the validation process is deterministic, because, you know they will they know the endorsement policy. So they will check whether the endorsement is being made properly, according to the policy and they will check the read/write

sets for the validity of the transactions. And then now a Fabric introduces a novel hybrid replication paradigm in the Byzantine model.

So you can have passive replication or and you can also have active replication. So passive replication is because pre consensus computation of state updates. So there is no real replication happening here. And active replication is happening when a post consensus, when everybody replicates the state change and everybody replicates the new block added to the blockchain.

And as you can see that in this scenario, there is no possibility of the blockchain diverging. Because after the ordering service creates a block the only that block is available to be added to the blockchain. And the so there is never will happen that there will be another block added to the last block and then there will be a race between two branches and then eventually one branch will win, all that stuff is gone in this scenario.

And then other thing is that in blockchain in the bitcoin and Ethereum, we saw that all transactions in a block must be valid, right. Otherwise the block will never be accepted. So when each node gets a block that has solved the proof of work puzzle, what it does is that it then checks first that the proof of puzzle solution given in the block is correct. And then it will check whether the transactions are valid by executing the transactions.

And that means that if it finds one of the transaction is invalid, then it will basically tell you that this block is invalid. So then anybody else who has solved the proof of work puzzle, that block will eventually come and then that will be accepted. Now since proof of work is so expensive, nobody actually in their right mind will create a block with invalid transaction because they are also executing the transaction to check validating before they cut the block.

So if I have put so much expense, computational expense to solve the proof of work puzzle, I better not put an invalid transaction because I know that peers will again execute the transactions and they will find that it is an invalid transaction, right. So that is the incentivization mechanism that works in case of bitcoin and Ethereum.

Because if you cut a block with invalid transaction, you are going to lose the money that you would have earned by you know your block being rejected. Since here there is no native currency there is no way to incentivize. Therefore the ordering service or consensus has to be done using standard distributed algorithm concepts and algorithms and also then you have to consider Byzantine failure and what happens, how to solve the Byzantine failure problem.

So that basically brings us to the end of the Hyperledger discussion. In the next class what we are going to do is that we are going to show you two more different types of blockchain technology. One is called IOTA. IOTA is a blockchain that was particularly designed for IoT devices, because IoT devices do a lot of transaction among themselves because smart home and smart transport and smart factory etc., are supposed to have these IoT devices and they cooperate with each other.

So they have to do transactions among each other. So IOTA is a infrastructure where they use a particular type of data structure called Tango. And they have a very different idea of a blockchain. It is in fact, it is not really a blockchain in the sense of blocks. It is actually a directed acyclic graph. So we will look at that, you know not in as much details as we have seen this blockchains.

But to give you a flavor of what the other kind of blockchains that are being developed for very different purposes, we can discuss that. Second thing that I want to discuss is another blockchain called Corda, which is again a business blockchain. So in some way, it is more closer to Hyperledger. But in some ways, it is very different in purpose and design than Hyperledger.

But we will discuss Corda also to show you what the financial technology financial world, the banks etc., they are thinking in terms of a blockchain based smart contract system, where the contracts are actually the part of the design and language and there we are talking about financial business contracts. Not the contract in the sense that, you know I, if you give me this, I will give you that kind of thing.

I mean, it is kind of that kind of thing, but, but it is more formalized. And the blockchain is designed with a much more formalization of business contract recording as well as business contract execution. So these two blockchain examples will be given lot more briefly than we have been doing with this three these three major block chains. Because in this course, we wanted to give you a very good conceptual understanding of what blockchain technology is about.

What where it is going? What are the different types of blockchain, it is not all size fits all, same size fits all. So there are different types of solution for different purposes and they have very different design, different intent. And then, so these two will give you some more exposure to this kind of blockchain. Once we have exposed you to that those two, of course there are many more, right.

So there are at least now about 100 probably different types of blockchains. Now the question is, once you have gotten the this very basic ideas, you will be able to choose based on your application needs and your business needs, etc., what is the right type of blockchain that you might use. Finally, we will towards the end of the course, we will talk about some applications.

We talked about applications in a very abstract manner like you know we talked about, you know supply chain, banking, finance, monetary system etc. But now we will talk about a few concrete examples of the blockchain technology being applied to non cryptocurrency applications. And hopefully with all this, you will be in a position to actually have a very conceptual knowledge of blockchain technology and applications.

And the ability to articulate what your problem requires if you want to have a blockchain solution. Also you know you should be able to articulate when your application does not need a blockchain. So we will discuss all that in the last week of classes. So we will see you next time.

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

Lecture - 25

Welcome to the seventh week of the blockchain technology and applications course. We are recording in a very strange situation. The whole India is going through a lockdown and our staff who actually to our studio recording are unable to come to work. So we are going to record this ourselves. And therefore, you will see that you would not see the instructor, but you will hear us and you will also see the slides as we go through them.

So with that in mind, I would request your indulgence with this form of presentation where the instructor is invisible, but hopefully that will not take away from learning. So in the past few weeks, we have been talking about cryptocurrency blockchains. Then we talked about a non cryptocurrency blockchain namely Hyperledger. And we discussed the differences between permissioned and permissionless blockchain.

And in case of permission blockchain like Hyperledger, we have seen that we have to have some kind of an identity and therefore, the blockchain had an Identity Service. And so as a result, the huge burden of making consensus among the nodes that are permissionless that are kind of anonymous is you know fulfilled by a very computationally expensive process, namely proof of work.

And in case of Hyperledger we saw that, that is no longer required. And in fact, Hyperledger has a notion of validation first and then it has the ability to order the transactions. And then it has the ability to actually you know put the information in the ledger after making sure that the transactions do not conflict. And that is how the Hyperledger blockchain worked. Now today I am going to talk about two different block chains.

One is specifically designed to work in a particular technological situation that is arising out of the extensive use of the Internet of Things or IoT devices, which are autonomous devices, which communicate with each other. And therefore, they often rely on each other on gating services and therefore, and the shared resources. And therefore, there is a notion of payment in terms of getting some services done by one device from the resources of other devices.

And to create such a payment in a way so that the all the payments and all the requests for resource usage or request for computational resource borrowing, one has to actually create what we call transactions and these transactions will obviously have a cost associated with it. And because of that cost, we have payments.

And because we have payments, of course, we could have done payments offline, which would mean that we would actually have a separate offline channel through which the owners of the devices could actually bill each other and have settlements.

But that is not very effective, especially given that the number of such devices and number of different owners could be daunting. And therefore, a blockchain based solution was proposed in the form of a specific technology called IOTA. Now this technology relies on notions that are German to the blockchain. But as we will see that they are not really using exactly the blockchain technology as we have seen so far.

They are using a quite different data structure. They are using a quite different method of validation and consensus. And therefore, some people would consider IOTA is as not a really a blockchain framework, but because of the similarities with the underlying technologies that they use in terms of digital signature, hashing, validation, etc., consensus, so therefore, we would consider it as a specific, customized blockchain technology.

So when I say customized let me remind you that bitcoin for example, is also somewhat customized for bitcoin. But as we have seen that somebody could develop other blockchains, other kind of solutions on top of the bitcoin blockchain. We know that we can persist some information for timestamping, or making certain information permanent and verifiable with the help of bitcoin blockchain.

For example, an educational system might decide to have their students graduation certificates in the blockchain using a you know using a particular field that we talked about, where we use some kind of a proof of burn kind of transaction to persist information. We could also develop other tokens on top of the bitcoin blockchain. But in general, the bitcoin blockchain is mainly known for a specific cryptocurrency called bitcoin.

So in that sense, it is quite custom type developed blockchain technology. On the other hand, the Ethereum decided that, that is not very useful in the sense that if bitcoin does not become the currency of choice of people, then the bitcoin blockchain technology will also not find much use and therefore, it might not survive. But Ethereum is designed in a much more generic way so that using smart contracts, you could actually develop any kind of application.

So you can use it as a cryptocurrency application framework, but you can also develop voting applications, you can develop land record registration, you can develop solutions for medical information sharing and medical information

integrity and security provider. You could develop solutions for certification, you can provide solutions for creating identities.

You can create solutions for doing providing certificates, like even the digital certificates to people. So in that sense, we know that Ethereum is a very generic blockchain framework, which can be customized by developing suitable DApps to solve many different types of problems. The Hyperledger on the other hand is also pretty generic. So it is not designed with any specific application or any domain in mind.

And therefore, you can develop many different kinds of solution. Even you can develop a some kind of a digital coin mechanism for as a currency on top of Hyperledger. So the one that I first want to talk about, as I already mentioned is the particular blockchain technology developed for Internet of Things ecosystem. I see it as a customized solution for solving problems arising in a very specific domain.

And therefore it has its own form and function that can also be, you know somehow molded into other applications. But it is not as easy to mold it into other applications as it would be for something like Ethereum or Hyperledger. So that is one important thing to remember. The other one that we are going to talk about is the Corda, which is also designed for financial players like banks, the exchanges and the various kind of financial you know players in the finance market.

And therefore, it is also quite customized and designed specifically for enforcing and executing the provisions of legal contracts that happened between financial Institutes. And therefore, their design and their architecture are very much customized with those kind of applications in mind. Having said that, that does not mean that they cannot be molded into something different, especially if they have the flavor of legal contracts and the contracts are to be executed over time.

But still, it is customized. So therefore, today's class, mostly will focus or this week's classes will mostly focus on this kind of customized blockchains.

(Refer Slide Time: 11:58)

So let us first talk about Iota. So Iota is a blockchain technology, actually the underlying blockchain technology is called Tangle. Tangle is the basic data structure and basic you know protocol by which the transactions between autonomous agents that are running inside various devices, Internet of Thing devices happen and the overall system is done by the IOTA Foundation.

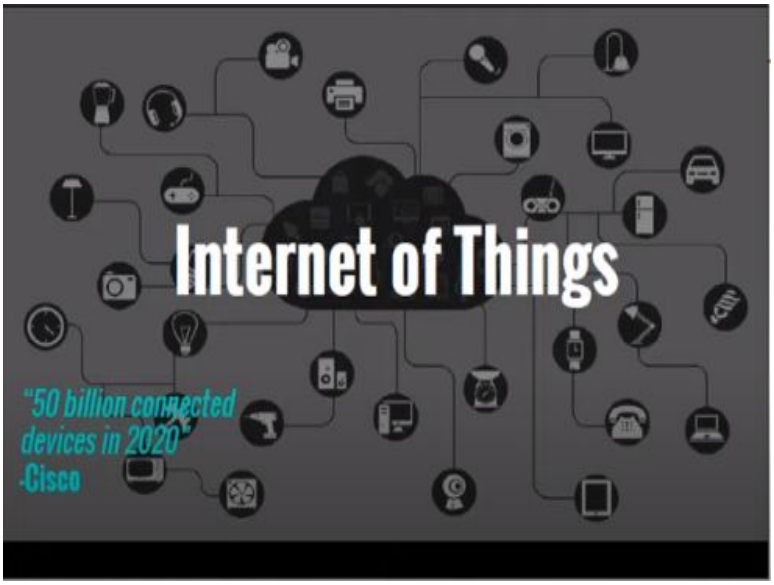
The name IOTA already has the Internet of Things in it. So you can imagine the reason for its creation. So most of the thing that I am going to talk about today are in the IOTA white paper which is quite widely available. And then certain blogs, for example there are blogs on the Tangle. And then there are certain simulations I am going to show that is also from this particular site here.

(Refer Slide Time: 13:31)

So let us see. So we are not going to let me see where I am, okay. So let us talk about IOTA. And in IOTA, there are three basic concepts that we have to understand. One is the concept of transactions. The concept of transaction

confirmation and consensus about a transaction being you know valid, no double spend has been has occurred. And the digital signatures associated with the transactions check out. And this basically are things that are has to be checked as part of the consensus.

(Refer Slide Time: 14:15)



So what is internet of things? So all of you probably know what Internet of Things are. So we talk about smart home these days, right? So smart home has smart you know thermostats, it has Smart TV, it may have a smart nowadays smart refrigerator, smart microwaves, it may have smart lighting, it may have smart you know smart grid components like rooftop solar panel, and various ways to control the various electrical equipments to save energy.

And all those things require a lot of sensors and lot of actuators and control. So in order for these things to work together to cooperate towards a goal, for example a smart HVAC system or heating and cooling system would be to have sensors in each room. And then based on a certain goal of let us say, minimizing the expenditure on energy, the different room temperature sensors and room, you know ducts and vents has to be opened.

The temperature sensor has to send data to a controller, the controller may actually separately control the needs of various rooms and open the vents with larger aperture or shorter aperture, this kind of stuff. So that what that means is that all these devices communicate to a controller, maybe they also communicate with each other in order for achieving certain goal.

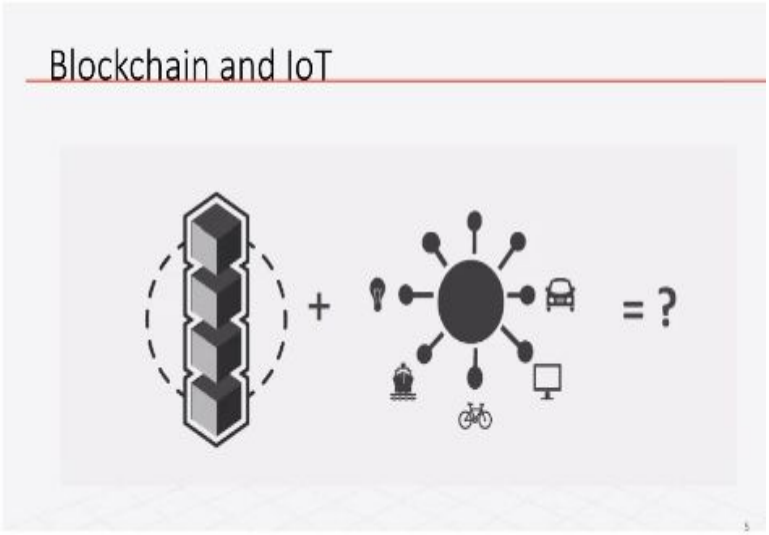
And I would call it a computational goal or kind of information technology goal, which might be associated with the cost optimization and other requirements. Similarly, you can talk about smart grid. Smart grid is the power grid, which is more and more having lots of communication between various devices.

Now these devices are not as simple as the smart home devices, they are often quite complex devices. They have many controllers at various locations of the grid, and there may be a overall supervisory control. And all these things together, they actually make a whole smart grid. And similarly, you can have a smart transportation, you can have various kinds of other you know device cooperation based infrastructure.

For example, vehicles like smart vehicles, they actually talk to each other, in order to avoid accidents, to understand the location of each other, especially around corners. They actually may talk to the infrastructure on the side of the road, they can even talk to infrastructure within the within the cars, these kind of things. So Cisco had earlier predicted that there will be 50 billion connected devices in 2020.

Now this is an older prediction. Now that we are in 2020 we probably have such numbers because the smartphones themselves are often part of the Internet of Things and then we have you know smart grid, we have smart manufacturing systems, we have smart transportation, all kinds of smartnesses that we talk about this requires many devices and sensors.

(Refer Slide Time: 18:21)



Now so what is the role of a blockchain in all this? So as we go forward with the technology of Internet of Things, it turns out that many of these devices are not necessarily very computationally resourceful. So but they need to do some computation, especially if AI is thrown into the picture for them to actually learn from the environment and then decide on certain control actions etc.

So then they can actually borrow resources, computational resources, storage resources, bandwidth etc., from other devices and therefore, they may have to transact with other devices. So note that there is no human intervention involved when they do that. So now the question is should now these devices are often not owned by the same company or same person.

And therefore, why should a device owned by another entity, you know serve some purpose for a device that is under the control of a different entity. So there has to be some kind of business relationship. In order to have that business relationship, what they have to do is they have to request a service and in return of that service they might have to pay.

So now for example, a cars that are electric vehicles, and they need charging. So there are roadside charging stations. They go to the charging station and get charged. So they have to pay. Now this payment could be done in traditional manner by keeping track of the amount of resources with the timestamp, you know like how we use credit cards and have the transaction log and have monthly billing or periodic billing of the owner of that particular device or particular infrastructure, you know.

And then bill comes to the user, in which case the user is the owner of the device. And then they can actually pay by banking and cheque and whatever. That is an old school method. But in this method, there are several problems. One problem is that these transactions you may get over built or you may actually somehow some transaction may be missing and you may be under built.

So and then somebody the owner, who is keeping all this transaction data going to him from his device might actually tamper with the transactions and over bill you. So these are some of the human problems, but the bigger problem could be that if we are having billions of such devices, obviously not all these devices are owned by the same entity. And many of these devices are, you know maybe in neighboring locations.

But they are owned by entities that are not even familiar with each other in the real world familiarity sense. So therefore, it is arguably a good thing if this devices can complete the monetary part of the transaction also unto themselves and we do not have to involve human in human beings to do the billing and other activities. Now how could device pay another device?

And the only way the device can pay another device is to have some kind of a digital currency. So obviously, when you think about digital currency in today's world, we think about cryptocurrency. So if you have cryptocurrency, that

you have put in the wallet of a device, then the device as it needs, it can negotiate with another device completely autonomously and make the payment and get done with the transactions.

And then eventually that cryptocurrency that goes to the wallet of the other device, that can be either used for it to actually spend on acquiring resources for its other computational reasons or other activities. Or eventually its owner could actually somehow encash it through some exchange. And that is where the IOTA cryptocurrency or IOTA coins come in to help, come to help.

Because now if you could create a cryptocurrency system, then we can actually have this thing completely autonomous without any human intervention. Also all this transactions that happens, transaction would mean that I am requesting some resource and I am going to pay this amount and somebody else is using that amount for doing another transaction to some of its other neighboring devices and stuff like that.

And this kind of information of the transactions could also be persisted in a blockchain so that later on some auditing and other kind of AI based usage of that transaction history to optimize the location of the devices, optimize the amount of resources available to the devices, etc., can be also done by the owning entity for all the devices. So therefore, blockchain seems to be a good solution to do this.

And as I said that if it was if all the devices were owned by one entity, then you probably would not need a blockchain and you could probably do this either free of cost for device owned by you helping another device that is also owned by you or it could be done through some centralized you know server, in which all the data, transaction data is persisted etc.

But here we are talking about entities that do not trust each other, do not know each other and they are still doing this completely autonomously. Now the question then would be that why cannot I use bitcoin? Or why cannot I use Ethereum, the Ethereum Mainnet to do all this transaction. And ideally, you could. But the problems are that first of all these cryptocurrencies are pretty expensive.

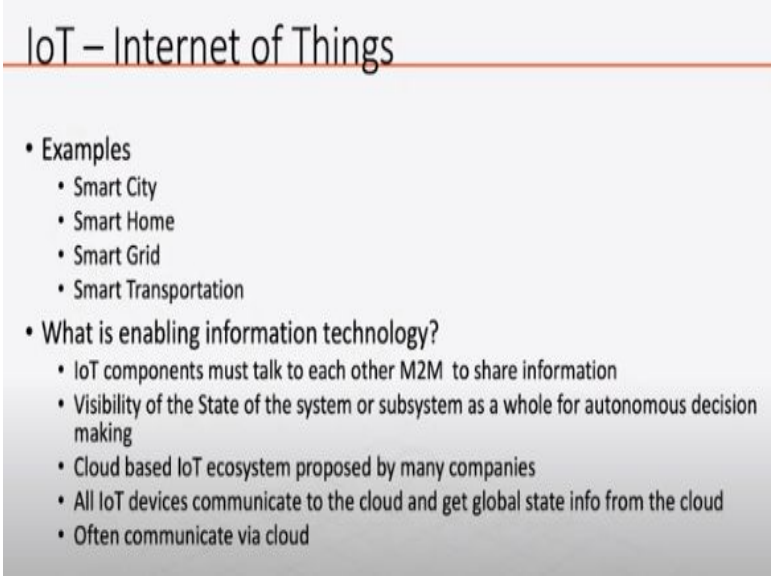
So that is one issue. But the second issue is that the, in order to interface these devices with such blockchains, they have to become nodes in the blockchain, this devices because you do not want any human intervention. And therefore, you want them to become nodes of this blockchain.

You could probably do a little better and you could probably have a sort of like a mediator which actually mediates for multiple different devices and then be part of this node, but then the granularity of the transaction history etc., will be lost, because any kind of manipulation of the transactions of these group of devices can be done by this mediator.

So the other thing is that the if obviously these such devices are resource constrained, so they cannot become nodes to this blockchain, which actually are or have millions of, you know transactions and blocks. And then finally, the other issue is that the transaction speed for bitcoin and Ethereum are not enough. So if you have billions of devices, and they need computational help from other devices, we are talking about a timescale of milliseconds.

Whereas these things actually have timescales of seconds to minutes, so they do not match properly. So therefore, it makes sense to actually consider a different blockchain solution. And that is what IOTA was trying to solve.

(Refer Slide Time: 27:10)



So going back to what I was already telling you about this internet of things, I already mentioned that smart city, smart home, smart grid, smart transportation, these are places where this IoT ecosystem thrive. But so what are the enabling information technology? So first of all, as I mentioned, IoT components must talk to each other machine to machine or M to M to share information, request resources, make payments, etc.

The state of the entire system or a sub system a group of devices as a whole has to be somehow visible to somebody, right? Obviously, each device may not keep track of the entire state of the entire ecosystem, because they are resource constrained, but somebody has to keep track of the state and then make autonomous decisions. It may be done together by all the devices, but not by some of the individual devices.

Now people have actually talked about or have implemented cloud based IoT ecosystem. For example, Azure IoT by Microsoft is one example. And if you do a search on Google as your IoT, then you will find out that they actually provide similar solutions. So they will allow to connect all your devices to an application hosted on Azure and that application will get data from each of the IoT devices.

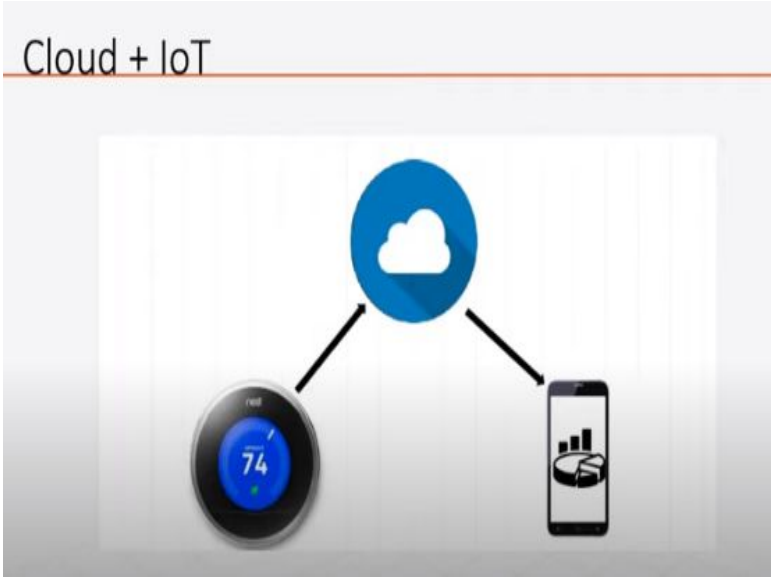
And then your application can infer the entire global state and make decisions and send the decisions back to the individual devices to adjust maybe, for smart valves, they can ask them to adjust the luminosity or adjust to turn them off. It can actually reduce the speed of a turbine. It can reduce the speed of a ceiling fan or it can reduce the temperature of a refrigerator or it can change the air condition setting etc., etc.

So often this things are communicating via the cloud. Now as I said before, this cloud based solution would work well, if you own all the IoT devices. For example, in your smart home, if you have multiple devices, you can actually use Microsoft Azure IoT or many other such cloud based offerings and have everything control via the application hosted in cloud.

But if you are going to not own all these devices and there are you know hundreds of different entities that own all these devices, then you cannot be trusted to have access to all those devices, state information as well as controlling them, right. So therefore or resource sharing decisions, you know on a single cloud. In that case, the cloud has to be managed by multi party.

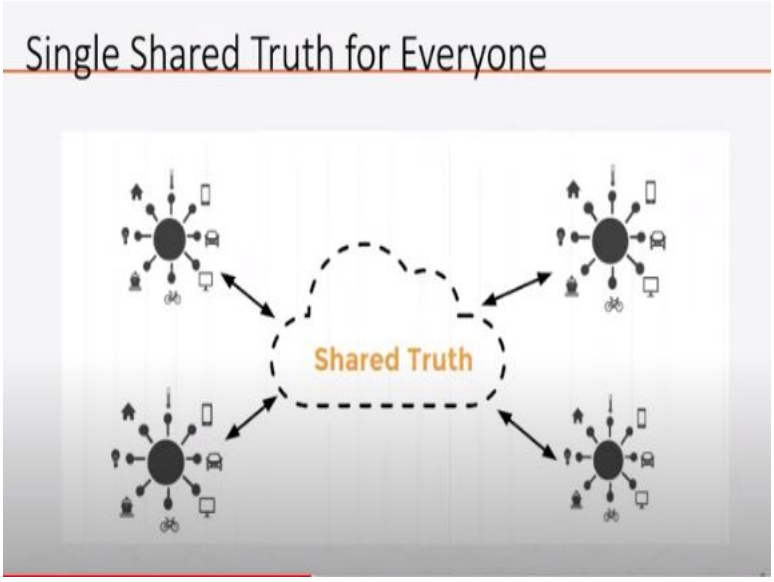
And that would basically give us bring us back to the situation of a blockchain which is basically a decentralized application rather than being hosted in a centralized place like a cloud.

(Refer Slide Time: 31:04)



So this is kind of the scenario I was talking about. So if you have a thermostat, and if you have a smartphone to control that thermostat remotely, then this thermostat might be uploading all the information to the cloud. And then your smartphone is also interacting with the cloud. And the smartphone, when it wants to control the thermostat, it tells the cloud then the cloud then tells in the thermostat to actually change its setting. So that is the kind of scenario we are talking about.

(Refer Slide Time: 31:39)

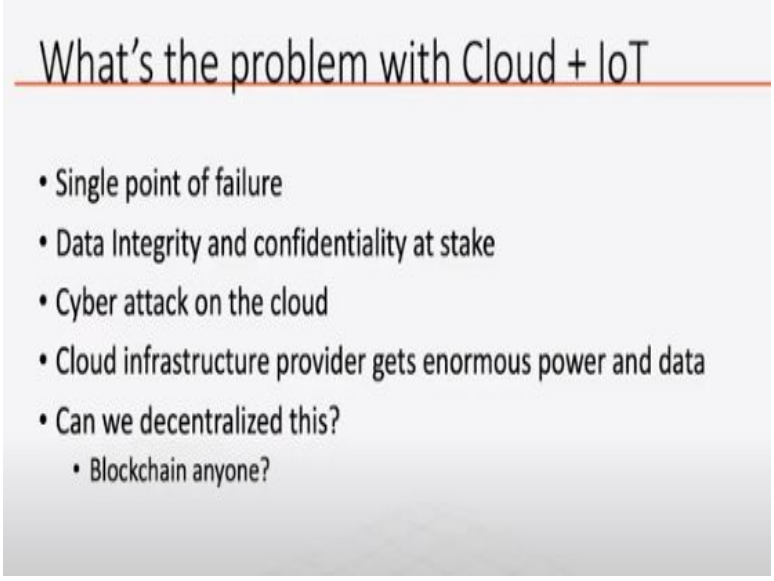


So what we in all this kind of cloud based solution, or if you want any other kind of solutions, in all these cases, all the devices should have the single shared truth about everything, right. Now it may or may not store the single shared truth in itself or in a cloud, but it has to be accessible to them, right. So you may actually have the shared state of the entire system with all the devices, current state etc., in the cloud.

That does not mean that each of the devices is looking it up. But if they want to look it up, they can actually look it up. And that is the whole idea of having this having a solution for this IoT, you know system control and decision making. If the shared truth is different at different points of the system, then they will make inconsistent decisions. And that is not good for you.

And we know already from this class that the blockchain is a repository of single shared truth, because as we have seen that through consensus mechanism through validation we always have the shared truth that is true, that is known to be true by everybody who is participating in the framework or in the system. Right. So that is the, another way of looking at the need for a solution where blockchain seems to fit well.

(Refer Slide Time: 33:25)



So I already discussed this that the cloud based IoT, you know decision making and control mechanism has several problems. The cloud is a single point of failure. And in that sense it is not just a failure in terms of that if the cloud crashes then the entire system cannot be cannot proceed. But also the cloud is a point where somebody is managing the cloud and therefore data integrity maybe questioned.

Like so some devices owning entity might say that the information that is kept in the cloud about its devices may have been tampered with. Or he or she might not even want that the all the information about this device be visible to the owner of the cloud or it may be made visible to all the other stakeholders in the system, which is not necessarily desirable.

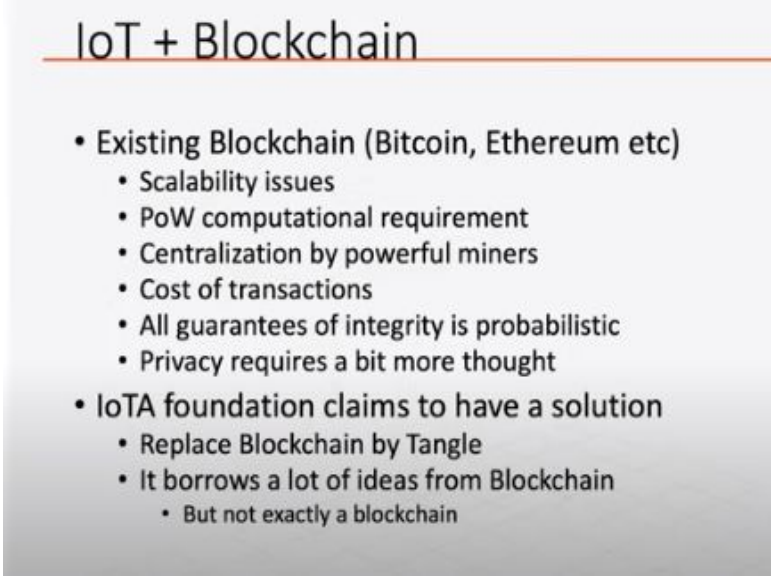
So integrity and confidentiality could be at least a question. If not compromised, it may be questioned. And there is no way to argue that, that has not happened without an explicit mechanism to prove integrity and confidentiality. And

again, that brings us to blockchain where these things could be actually arranged to be proven, or by construction, it has to be trusted. The cyber-attack can happen on the cloud.

Obviously, then again, the integrity and confidentiality would be at stake. And then also if a DoS attack happens or denial of service attack happens, then it may be even worse, and the cloud system would actually stop working. And therefore the entire the IoT ecosystem, the so called smart home or smart transportation will stop working.

So in that sense, the cloud infrastructure provider, like let us say Microsoft Azure will get enormous power and data over all the entities that are in the participating in that smart, whatever smart transportation, smart home, smart building, whatever. So we want to decentralize this. And certainly, decentralization after doing this class should bring to your mind the idea of using a blockchain.

(Refer Slide Time: 36:14)



So let us consider existing block chains like bitcoin in Ethereum. I have already hinted that these are not necessarily the best solution for the particular problem we are trying to solve. Scalability is an issue, right? If we are talking about 50 billion devices, of course not all devices would be using the same infrastructure. But still, if you have a smart transportation, we could think about hundreds of thousands of smart devices.

They need to interact, they need to make this they have a shared single truth. They need to have a decision making process and of course these blockchains are may not scale for that. The proof of work computational requirements of course, none of this smart devices are computationally endowed enough to actually take part in that.

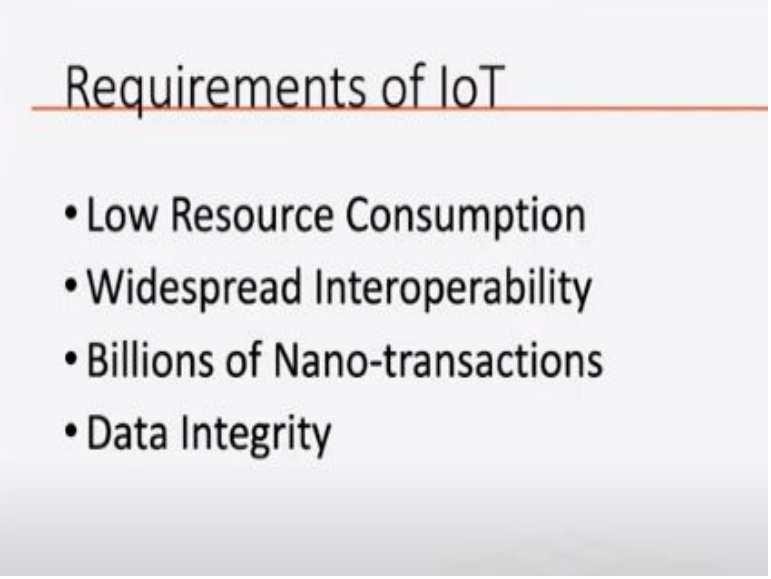
And the so therefore, there is a centralization tendency in bitcoin and Ethereum and we discussed this in the context of Ethereum when we talked about Ethereum that even though the decentralization and democratization was the goal of bitcoin and Ethereum, but turns out that you know 90% or even more percentage of the transactions, the money etc., are either held by a very few of the nodes.

And that basically would mean that it is going back to the centralized solution kind of thing which we do not want and that is why we said that cloud is not the most desirable solution to this problem. The cost of transactions of course, will be very high because all this cryptocurrencies are valued at a very high value.

And the guarantees of integrity is obviously probabilistic, which we cannot avoid even in this case, but that is a that is an issue. And then privacy would require more thought because as it is, bitcoin or Ethereum are not encrypting any of the transactions or any of the state information, and therefore everybody can see everything in those blockchain all the past history, and transactions and etc.

In our case, we may not want that kind of thing. So the IOTA foundation came up with this notion of IOTA and they replaced the blockchain by this kind of a structure called Tangle and it borrows a lot of ideas from the blockchain, but it is not really a blockchain and as we will see very soon.

(Refer Slide Time: 39:24)

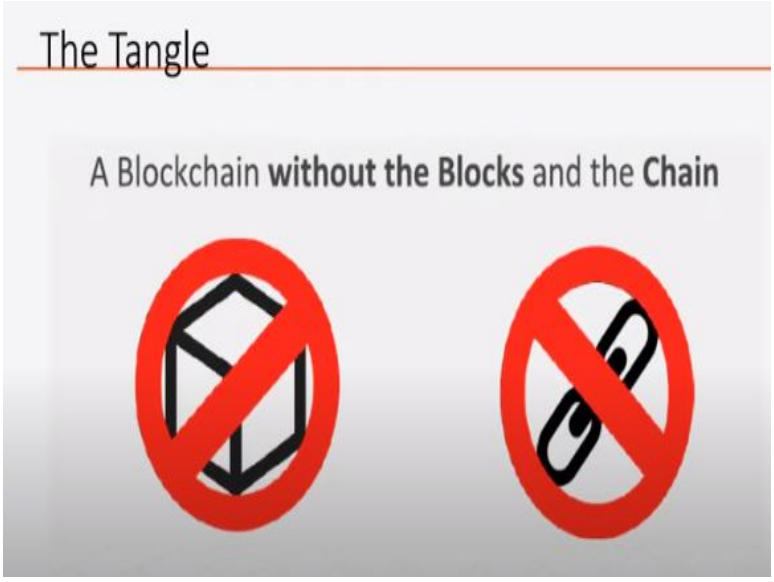


So requirements of the IoT devices to be part of something like a blockchain is that the resource consumption for participating in the blockchain other than its own actual functionality has to be low, right. And then there has to be interoperability, right. So you cannot have for, you know some custom solution for specific manufacturers of IoT devices.

It has to be completely interoperable among all devices, among all manufacturers, etc. And the number of transactions is mind boggling. And that is one of the reasons why bitcoin Ethereum etc., cannot cope with this is there will be billions of nano transactions. And then data integrity is important as with any of these things, so we do not want the data to be left just in a database, all the transaction data.

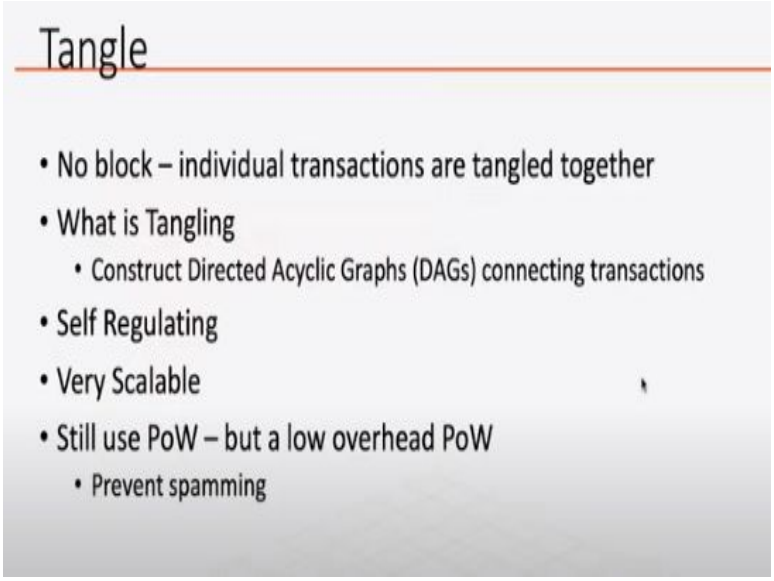
But it has to have some integrity mechanism against any kind of manipulation of the data.

(Refer Slide Time: 40:41)



So as I said that IOTA is actually introducing this structure called Tangle, which is a blockchain without blocks and without chain, right. So what could that be?

(Refer Slide Time: 41:00)



So in Tangle, we have no notion of a block. So all the individual transactions actually are called are actually nodes of a data structure or a graph. So visualize bitcoin blockchain or Ethereum blockchain as a chain of nodes, right? So each node contains a block and the block contains many transactions. And then we have a chain of those. That is what we see as bitcoin or Ethereum etc.

Whereas here, we are only talking about individual transactions as nodes. And they are connected with each other through a graph structure. And we call we say that they are tangled together. And the idea is that transactions has nodes and they are connected by a directed acyclic graph. So what is a directed acyclic graph? All of you know what is a graph. Graph is a set of nodes and edges. Edges is what connects the nodes.

Acyclic graph is a graph in which nodes are there, edges are there, but there is no cycle, right. And then a directed acyclic graph is a directed graph which means that from node A to B if there is an edge, then there will be an arrow from A to B and but does not mean that B to A you can go because the arrow is unidirectional. If there is a way to go from B to A also then there should be edge or a path from B to A.

So unlike undirected graphs where having an edge between A to B means that I can go from A to B and come back from B to A, directed graphs are directional. So you cannot just go like that. So this is the kind of graph we are talking about. So instead of a chain, as in case of a regular blockchain, we have a graph which is a directed acyclic graph. Blockchain, the bitcoin blockchain also is directed, it is a directed chain, right.

So coming, starting from Genesis block, it is unidirectional, right. So this block, then Genesis block, then the next block and next block and so on. Here, it is a directed acyclic graph, but it is a directed graph. So Tangle is self-regulating. We will see what that means. It is quite scalable. And it still uses proof of work, but the proof of work is not so actually this should be corrected.

We meant the low overhead and not the long overhead. Low overhead proof of work. And the proof of work is mostly used not for consensus but to prevent spamming. And we will see why that is.

(Refer Slide Time: 44:26)

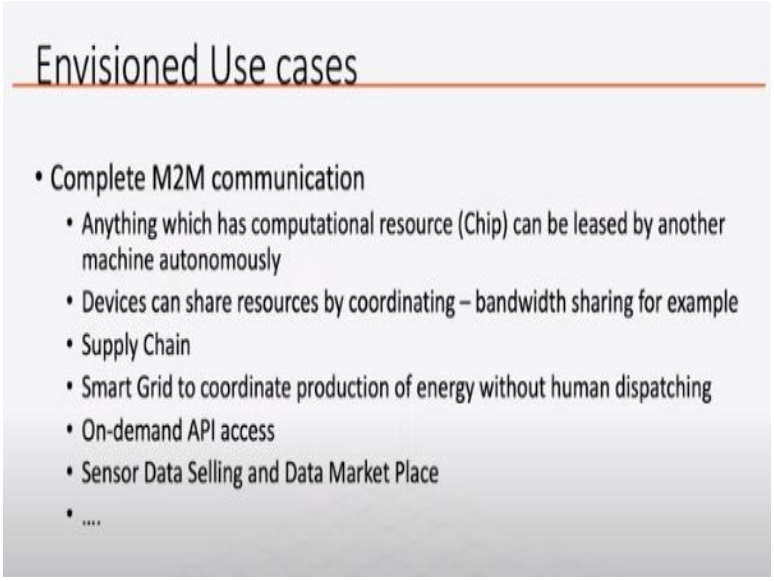
So what we get out of Tangle instead of using a traditional blockchain, right. So first of all there is this notion of Brewer's CAP theorem, which requires a distributed system to have consistency, availability and partition tolerance. So consistency is basically obtained by some kind of a notion of a consensus that the data at each node or notion of the shared state space at each node is the same consistent view.

Availability means that it should be available. The service or whatever service it is providing should be available all the time, which means there has to be some notion of fault tolerance of some kind and partition tolerance, and that is the P of the cap. Partition tolerance means since it is a decentralized system, so if some part of it becomes offline, even then the system should be able to work.

And we will see that the CAP theorem of Brewer is actually pretty much fulfilled or satisfied by IOTA. There is no fee involved in this tangle. And then it is scalable. It is modular. It is modular in the sense of partition tolerance. We will see that it is quite lightweight. And offline transaction is allowed. Actually modular, offline allowed partition tolerance, these are actually different facets of the same thing.

And then the cryptography to users is actually more quantum resistant cryptography, then what we saw in case of a bitcoin or Ethereum.

(Refer Slide Time: 46:24)



So what are the use cases for this? So as I said, the use case that is more close to my mind is that borrowing computational resources from one machine by another. So if I want to have a computation, I do not have the computational resources. I offload that to another machine which has the computational resource and instead I pay. So bandwidth sharing, for example, I could, I may need to urgently send some information.

And I do not have the bandwidth. So I ask a neighboring node to actually use its bandwidth to deliver or broadcast or communicate that information, which is same as computational resource sharing. It can also be used to keep track of supply chain, in case of smart grid to coordinate production of energy without human dispatching, on demand API access.

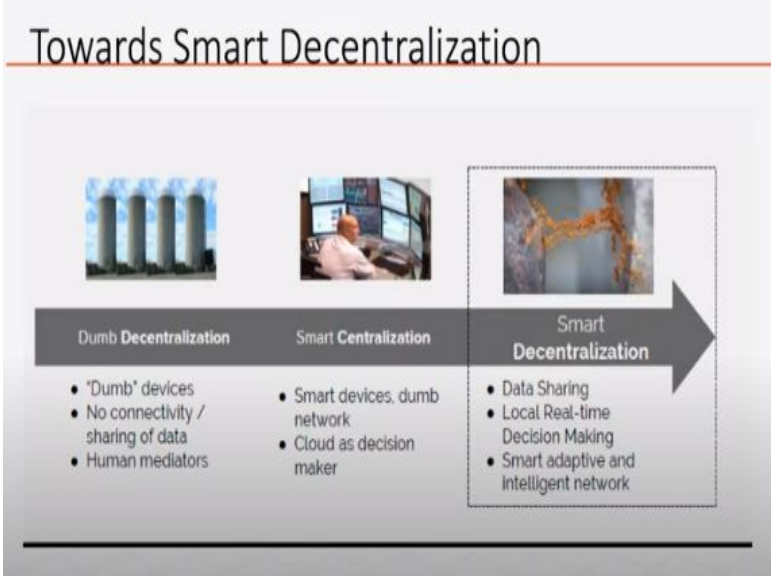
So I have a device which needs to use some API let us say for example, in most cases, I can envision if this devices are also using some kind of a machine learning based decision making, then they might not have the ability to call on an API directly on their own resources for let us say, calling a machine learning model, but they can ask another higher resource device to do that for them.

And then you can also do various kinds of data, business data marketplace. Devices collect a lot of data. And you may want to monetize that data, which is not happening all that much now but it is going to happen a lot more. So devices, which are collecting, let us say solar irradiation data in a smart solar panel system.

This data can be actually monetized and people actually do sell this data but not autonomously, like what is envisioned here. But more by dumping that data in a database and then the owner of the database can sell that data to

researchers and other companies that make solar related products. So this is the kind of things that we envision as the users of the IOTA system.

(Refer Slide Time: 49:15)



So decentralization, so few words about decentralization. So it turns out that in the old days, you have an industry and you have devices, but devices do not communicate with each other. They actually or communicate very locally. So they are not connected through a network. They may directly hardware to a controller, so they can send some data to a controller but not send that data elsewhere.

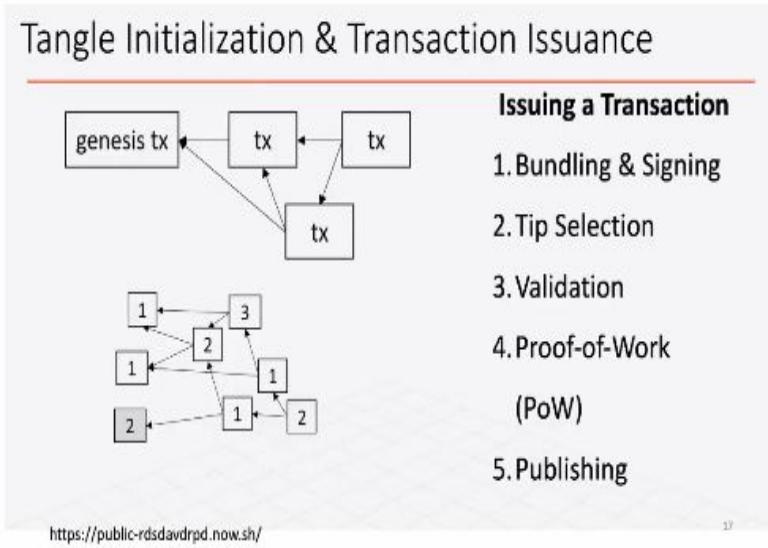
And most of the time, the human operators will use whatever data they are seeing, let us say, an HMI interface or human machine interface will then intervene to actually use the data collected by the device. So smart centralization came when smarter devices started appearing in the market, which can send its sensed information which can be sent actuation information.

And there the network does not have to be very clever. It is only a mediator through which this information flows. And all the information may actually go to a cloud and the cloud may do all the necessary computation for decision making and then send the actuation information to the devices. So that is a very centralized solution, or smart centralization.

So dumb decentralization was not very smart. So this is basically I would say, industry you know 2.0 or between industry 2.0 to industry 3.0. Smart centralization is more like industry 3.0. And then we are talking about smart decentralization where devices share data. There are local decision making, which is real time. And then smart adaptive and intelligent network. And this is actually what we are envisioning, as industry 4.0.

And this is where we will see this kind of a situation, the use cases that we were talking about in the previous slide, and this is where the IOTA could be very useful.

(Refer Slide Time: 51:47)



So let us now dive into Tangle, right. So in Tangle, there is this directed acyclic graph which looks like this. So you have, this graph starts with a node called the genesis transaction. And the genesis transaction actually, is where a pot of money, they are called the IOTA or IOTA coins, they are created. And then they are distributed through this transaction to a number of entities to in order to bootstrap the system.

So certain devices might actually be given some amount of IOTA, and IOTA, unlike bitcoin or Ethereum there is no notion of mining. So the first genesis, we create a pot of money, and that is all the money that it will have through its, you know life cycle, however long that life cycle goes, because money will keep changing hands and that is how the entire system will run.

And therefore, there is no way. Of course, one can also purchase from one of these devices or you know some of this through fiat currency through an exchange which accepts IOTA to fiat currency exchange, but that is all but there is no new IOTA coin that will be created. And then when transaction starts getting created, say devices want to make transaction they have to choose at least two previous transactions and validate them and then they add themselves into this graph.

So for example, this transaction cannot obviously have two transactions to validate because it before it, only genesis transaction was available. So it basically validates the genetic this genesis transaction. And when I say validate it

basically at this point, it only can validate the signature and stuff like that, there is not much to validate here. The genesis transaction might have given some money to various devices.

And it may might want to check that is actually not conflicting or something. But then a new transaction come in and this transaction has now two transactions which both of which it can validate. And that is what these two arrows mean that this transaction validated these two transactions.

Now another transaction comes almost concurrently and then this transaction chooses to validate this transaction which already has been validated by this transaction, but there is no harm in this guy validating this transaction and he also validates this transaction. So this is how the Tangle grows. As transaction come they look for two unvalidated transactions so far and then they validate them and then they attach them to those two transactions.

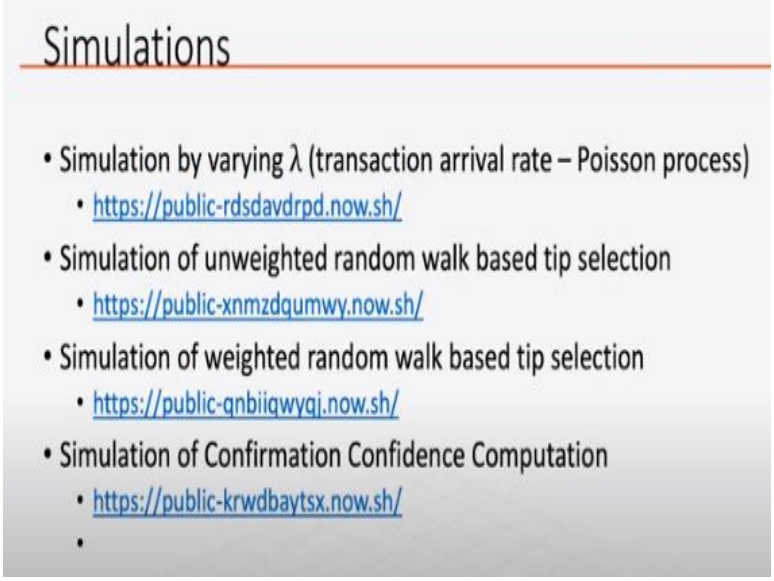
Now the transactions that they validate which are yet in not validated by anybody else are called the tips of the Tangle. So all the other transactions which are basically nodes in this graph, which has been validated by at least one transaction is called is an internal node whereas the nodes are transactions that have not been validated yet by any other new transaction are called the tips.

So and the transactions are actually bundled and signed by digital signature which is the done by the issuer of the transaction. And then, the as the new transaction comes in, it has to first do a tip selection. And we will talk about how this is done, the tip selection, and then we will talk about the validation, how they validate the two transactions or in some cases one transaction that it has to validate.

Now it also has to do a small proof of work. And this is because the, for a transaction to actually go into the tangle, it has to complete a small proof of work. And this proof of work is there in order to stop spamming. So what is spamming? Spamming means that a device may decide that I will create some meaningless transactions and just make the Tangle bigger, right? So it might go rogue, and it can start doing such things.

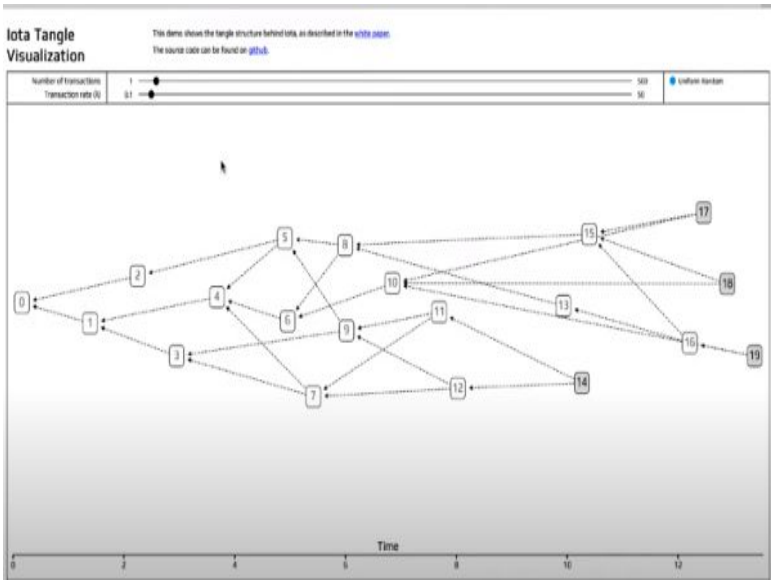
So we do not want that to happen. So therefore, we have to put some computational burden on the transaction created so that they do not start pumping out meaningless transactions. And then once the transaction is has passed all this then it will be published in the sense that it will be part of the Tangle and then anybody who has a copy of the Tangle, can see that this is this transaction has already made into this.

(Refer Slide Time: 57:26)



So there are several simulations that I want to show you. So here is the simulation of how a Tangle actually starts growing.

(Refer Slide Time: 57:44)

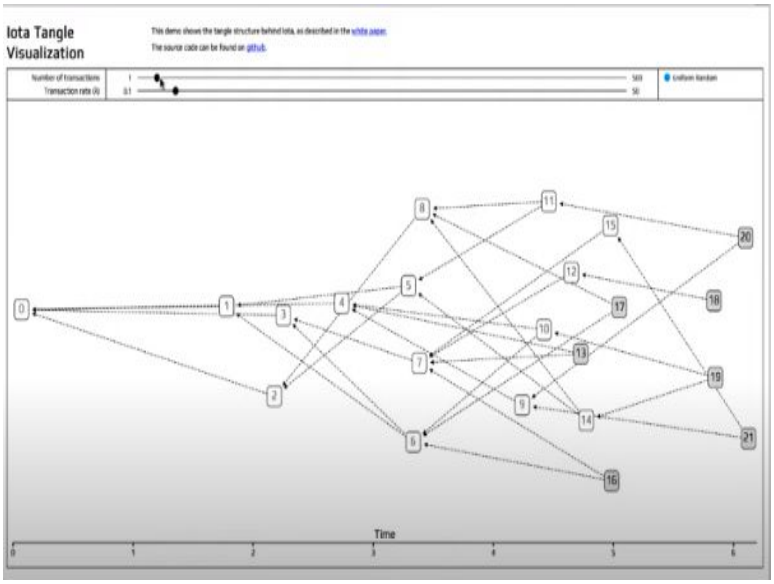


I do not know what is happening here. Okay, so here in this simulation, let us say the number of transactions is just one, right. And then I have a transaction rate, which is the rate at which transactions are created. You have many devices, each device may do its own transaction with another device, or you know they might just not do anything.

So here the transaction rate we have considered here is 0.1, which means that every 10 seconds, one transaction comes in. Now I increase the number of transactions from one to two, right? So the first transaction will find zero as unvalidated yet so it will validate it and do its proof of work and gets added. So now I have I add another one. Now 2 comes in and starts validating.

So it could have validated zero also but it decided to validate only 1. So here I do it another time. This is randomized simulation. So in this time it decided to validate 0. So now I have 3, now 4, 5, 6, 7 and this is how the tango starts growing.

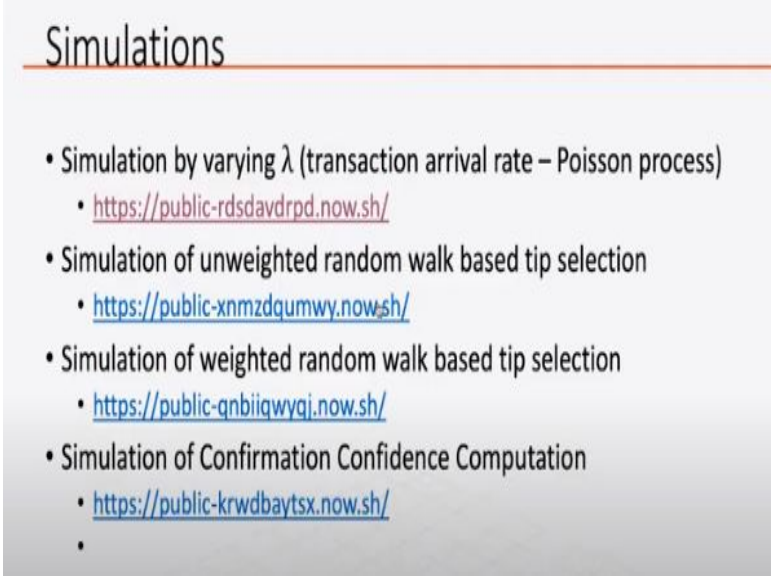
(Refer Slide Time: 59:37)



Now if I change the rate and I make the transaction rate very high, then the shape of the tangle will also change because more transactions that come in at the same time, if they come one by one, then they will find few tips. But if many transactions come in together then number of tips or invalidated transactions, they will find many. So there will be a lot of different variety of transactions to choose from for them to attach themselves to the transaction.

So the shape of their Tangle will depend on two factors. One is the number of transactions and the rate at which transaction come and this is how the transaction will grow.

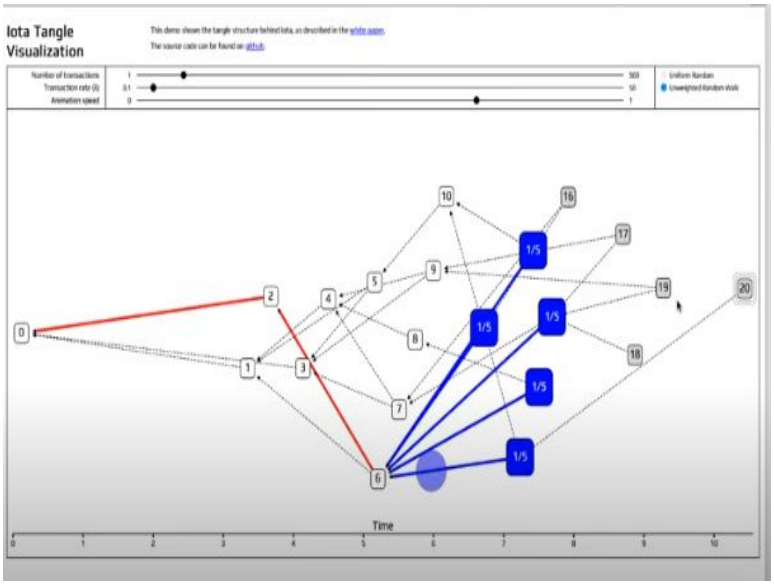
(Refer Slide Time: 1:00:28)



Now the second thing is that, how does it choose which tip to attach to, right? So there are let us say I have five tips and I only need two tips to select to connect myself to the Tangle by validating them. So there are a number of different possibilities. So first thing we would like we will look at is what is called an unweighted random walk based tip selection.

So what happens there is that when I am a new transaction, by the way I have access, I have access to the entire tangle that has been made so far.

(Refer Slide Time: 1:01:08)



So here, I have the automated animation, so the animation is happening. Now what you are seeing here is that what when a new transaction is coming, it starts from the genesis, it does a walk from the genesis block. And what it does is that in the in this unweighted mechanism, for a random walk, it basically gives equal probability. So it is in zero and it has right now three paths from zero.

So it gives equal probability and chooses one of them. Then it once it chooses 1, it comes to this node there it has, let us say 1, 2, 3 again probability. So each has one-third probability. Then it comes to this one. And this 1 has also two at least I see two outgoing, so it will have a half of probability of adding to the next one. So this way it will walk all the way from genesis to the whichever tip it ends in, after which it cannot proceed, because there is no more path. So then it chooses that tip.

And then it does the same thing again, so that it can choose the second one. So this is what is called the unweighted random walk. So unweighted random walk is a process by which the tip selection happens. But in this case, the tip

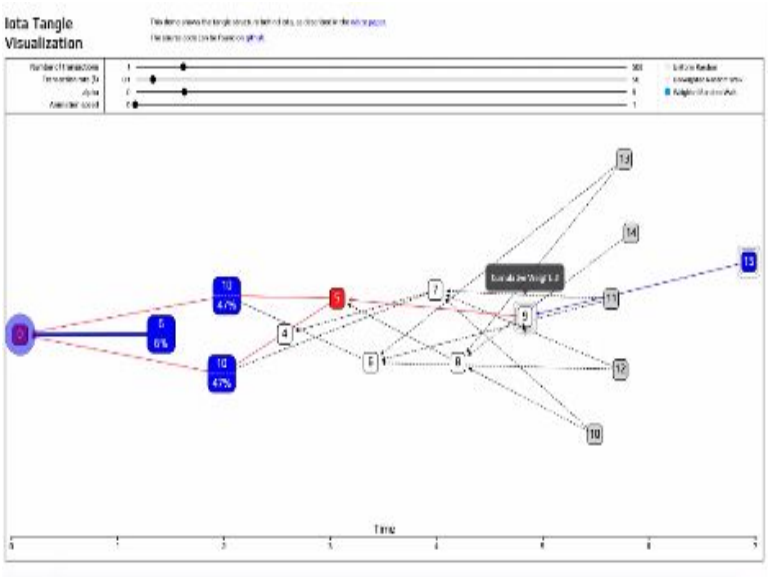
selection creates something called lazy transactions, lazy tips. Lazy tips are tips that basically goes and does not validate some new tips.

So a new transaction comes in and it does not do all the work. To find some new tips, it goes and does validate some already validated old transactions. And that is not desirable because we want that all the transactions to be validated. Now if every tip comes and validates, let us say zero, then every new transaction comes and validates zero, zero has already been validated several, by several node several transactions.

So it does not make sense to do that. So therefore, we have to have a way to penalize the new transactions that does this. So therefore, equal weight towards all directions at every intermediate node, while doing the random walk towards a tip is not a good idea, because then the lazy tips will also get validated with very good probability. So you want to reduce the probability that if a tip is lazy, it should not be allowed to be validated.

And therefore, if it never gets validated, the transaction never makes it to the permanent record. So that is what we want to do. So therefore, what we do is we do a weighted random walk based tip selection.

(Refer Slide Time: 1:04:23)



So weighted random walk based tip selection basically assigns different weights to the different directions, when it is on an intermediate path. So you see that here it is doing equal number of probabilities, equal probabilities. But now you see, there is a difference 33, 33, 33 and then as it grows more you will see we saw just 45,27, 27 and so on and so forth, right. So the question is that how does it decide what is the split, right.

The split has to be some algorithm right. So it cannot randomly say this is 23rd, 23% that is 38% and so on. So I have to have let us reduce the animation speed here. So you can have a better look at the numbers. So increase it a little more. So here you see 42, 42, 16. So when it comes it obviously will come towards 42, 73 and 27. So it will certainly high chance it is going here and so on and so forth.

And this is based on the notion of a cumulative weight. The cumulative weight of a node is based on how many transactions have already validated a particular transactions, and then that plus one. So we will discuss this lot more. And then based on the all the cumulative weights of all the nodes that are that you branch out to, you can then split the probability based on the in proportion to the cumulative weight of the nodes that you are connected to.

So here you see that this guy has a cumulative weight of 10, this has a 10, this has a 6. This guy has a 6 cumulative weight, this has a cumulative weight of 4. And this guy has a cumulative weight of 1, this guy has a cumulative weight of 4. So why is it has a cumulative weight of 4? Let me stop the stop this and look at this. So what is the cumulative weight here?

The cumulative weight here now is 2, because it has been only it has been validated by 1 and nobody else. Now 8 has been validated by 14, 13 and it has been validated by 10. So 3 validated it. So it has 4 cumulative weight. This guy has a cumulative weight of 7, because 2 have validated it, but those two have been validated by others. So 4 and so it has 4, and this guy has been validated by, so this is cumulative weight 4.

This is cumulative weight of 2. So 4 and 2 six. And then plus one is cumulative weight of 7. So this is 7, and let us say this is 5. So the probability between these two will be in the 7:5 ratio. So that is how the your tilting your path in the tip selection towards the nodes that have a higher cumulative weight because they have been already been validated by many more different new transactions.

So then finally, I want to show this animation which is about the what is called the confirmation, confirmation weight or confirmation score. So or confirmation confidence. So what is the confirmation confidence? Confirmation confidence is basically say the genesis node. It has been validated by couple of transactions right after they came after the genesis node.

But those nodes have in turn been validated by other nodes and those nodes have been validated by other nodes. And one thing I forgot to tell you is that when I validate a transaction, I have to make sure that whatever transaction that

guy has validated is also valid. So therefore I basically do an entire history of the nodes. The node I am validating or the transaction I am validating has validated some other transaction which in turn validated other transactions.

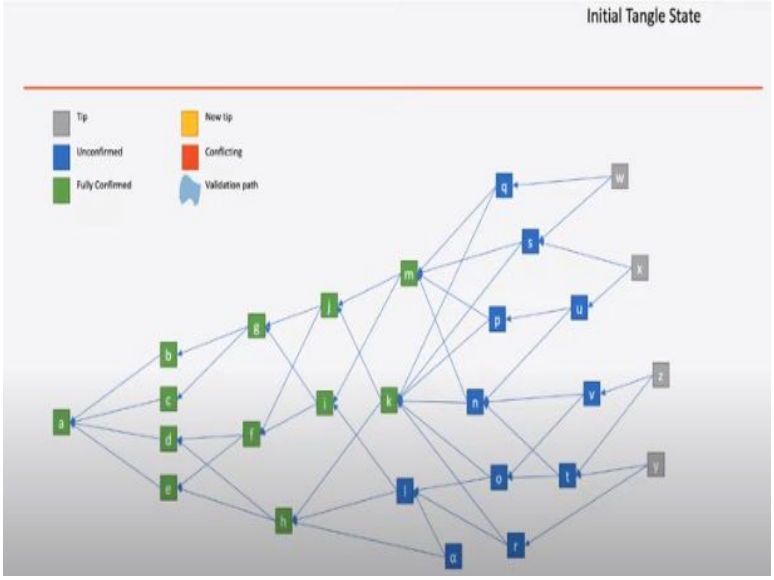
So every time I validate a transaction, I will not only validate the that particular transaction but any transaction that this transaction has validated and in the transitive manner. So therefore, the genesis node has been validated after some time. The genesis node has been validated by all the current transactions which has been attached to it. So the confidence that the transaction recorded in the genesis transaction is very high.

Whereas something that is in the tip which has not or close to the tip, it has been only validated by maybe one or two other transactions and not by very many others. So then I will say that the that one has not been the confirmation confidence is not very high. Now why do we need this confirmation confidence? That is because I may decide that say I put a transaction and I may want to do the result of the transaction.

So I got some money out of this transaction. I am a device and I got some IOTA coin, because I gave some resources to you and then I want to spend that resource for getting another resource. So I put another transaction. But that transaction, you know may or may not be trusted by others, because the previous transaction on which you depend on to put this transaction on is not necessarily very high of very high confidence.

So I have to wait until the high confidence increases. So now we will back to some more illustrative example.

(Refer Slide Time: 1:11:11)



So in this picture, so I have let us say this is the Tangle state in which I am looking at this. When I look at this and I see that some of the transaction nodes are painted green, and then some are blue and some are gray. So the gray ones

are tips. They have not been validated by anybody yet. So obviously, we cannot have any confidence in their confirmation because they have not been confirmed by anybody.

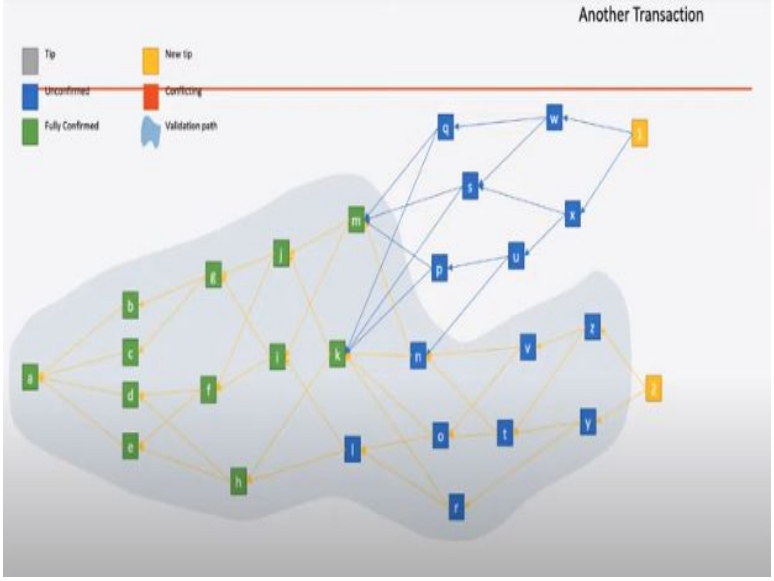
But the green ones for example b, b has been confirmed by g, g has been confirmed by j, which then in turn had to confirm b; j has been confirmed by m and m in turn confirm g and that in turn confirmed b. And then m has been confirmed by q. And that means in that process has confirmed v and w has confirmed q which in turn that process confirmed b.

So b has been confirmed many times and therefore, I would say that b is of high confidence. Whereas, or after a certain high confidence is reached, I will say that this is fully confirmed, whereas this blue ones, they have been only confirmed by few and therefore, they are not completely confirmed yet.

(Refer Slide Time: 1:12:45)

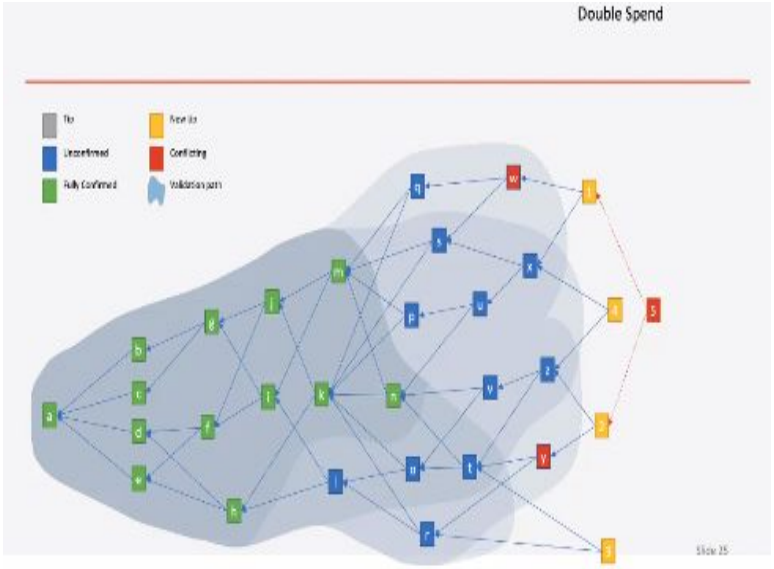
So when I add a new transaction now let us call it 1, which confirms w and x. And in the process, it conforms all the ones that are in this shaded region, because they somehow in turn has been confirmed by either x or w, right.

(Refer Slide Time: 1:13:11)



So now another transaction I and 2, which actually confirm z and y. In that case, what I will do is I will have this ones that had confirmed indirectly by z or y I will consider them as having a higher score of confirmation confidence.

(Refer Slide Time: 1:13:36)



So this way, I will keep doing this and if you take them to overlap, then you will see that the node n has been now added into green because it has reached the enough number of confirmations. So in my first picture if you look at n, n was blue, right. So only m and k were Green. But then after one and even after one has been added n is still blue, but then n has been now confirmed another time by addition of 2 and therefore n will now turn green.

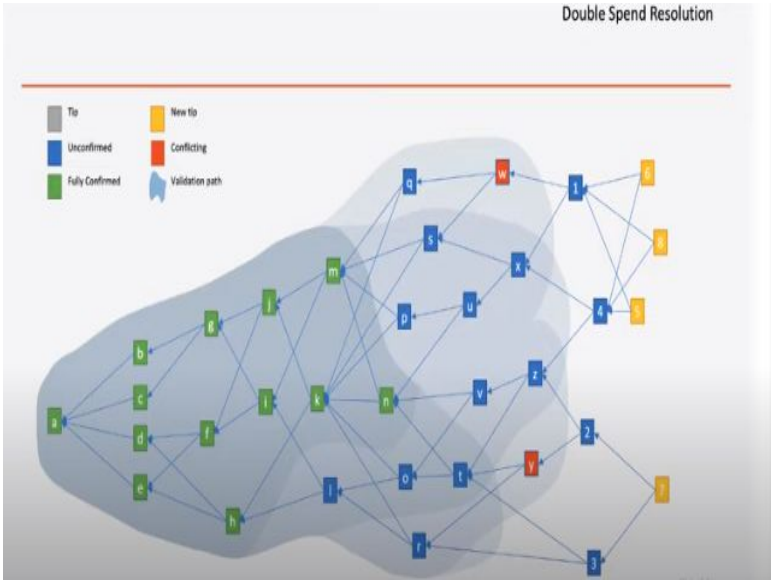
Now one issue is that. So this way if more and more new transactions come in, things get more and more confirmation, and therefore, their confirmation levels go up and eventually more ones turn green. Now there is one issue that can happen, which is the problem of double spending. Let us say w is spending a coin that it has gotten and from a previous transaction and y is doing the same thing.

And then 2 confirm z and y and 1 confirm w and x. So 1 did not know the existence of y similarly 2 does not know the existence of w. So they are double spending, but still has been confirmed at least by 1, one transactions. But this will not work very long, because soon after, as more and more transactions get added, some transaction will choose 1 and 2 together.

Or some successor of 1 and 2 together and at that time, they will they have to go to go through the entire history and then they will find a conflict w and y and therefore, w and y will no longer be in the path and anything that depends on w or only w may stay or y may stay. So that will depend on how people decide. So obviously 5 will obviously not continue to validate this.

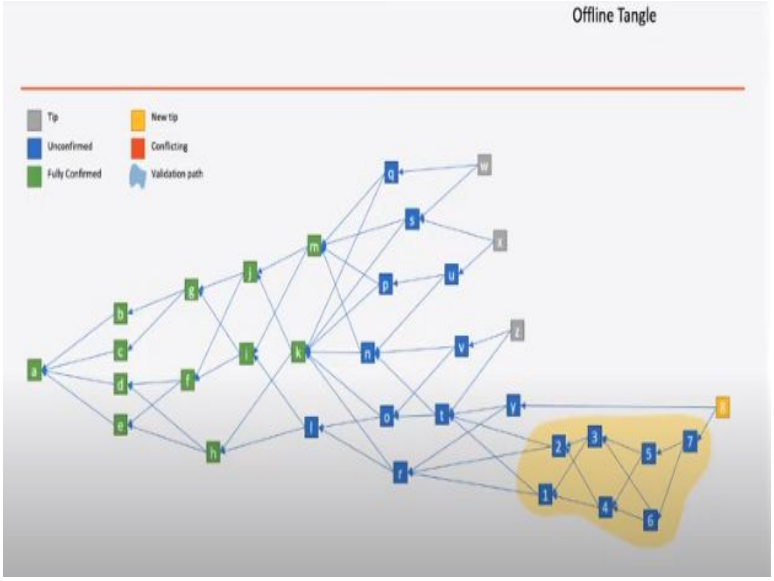
So 5 will then choose 1 and 4 maybe. And then this part the part of with 2 will get orphaned. So this will keep happening.

(Refer Slide Time: 1:16:15)



So this is how the confirmation level of y will not increase anymore, because after a while everybody will flock towards the upper side and then the in the lower side y will not remain very active. This part will not remain very active and therefore y will actually lose in gaining any more confirmation confidence. So this is how the double spending is resolved in Tangle.

(Refer Slide Time: 1:16:50)



So then the one last thing I want to say is that suppose I have a part of the network that gets segmented. So let us say I have a transaction. This yellow part that you are seeing is actually 1 and 2 started by having validation of t and r and but at that point they got disconnected from the rest of the network.

And then that does not stop its own partition of the network to keep transacting and in order to go back to the intertangled when the partition gets rejoined again, a new transaction has to come in to add to validate together the last transaction or tip of the offline transaction and tip of the online transaction together and then they will get connected.

Now if there is a double spend in this yellow region with the any of the nodes on the previously the online part, then after some time, that will be again be caught like we showed earlier between w and y. And that is how they then one of them will stop getting any more, stop growing and their confirmation confidence will stop growing, and therefore they will never become green. So that is the idea of having the offline Tangle.

So what we see from this is that the consistency in terms of resolving conflicting transactions, the availability in terms of having the ability to survive, making transactions in case of a network partition or becoming offline and the partition tolerance, that is if the network gets partitioned, the part of the network that is partitioned out, can still continue having the growth of transactions.

And later on resolving the consistency as it gets joined back in, all these things are satisfied. So CAP theorem in some sense holds for this Tangle framework. So that makes this one quite powerful. So this is all I am going to talk about IOTA because this is, you know one can of course go on having very long sessions on IOTA, going through how this

proof of work works here, how the what the transactions look like, what the and then maybe having some more coding examples, etc.

But that is not the point of this class. The class is to just give you the ability to see between different paradigms, even within the blockchain technology, the different paradigms, the difference between customized for a specific kind of application versus more generic type of blockchain. And then the fact that the blockchain is not necessarily always a chain, but it could have a different structure.

But it uses very similar concepts of transaction, transaction validation, some form of consistency making and in this case, even further partition tolerance and availability. So all this together gives you a very good sense of a different paradigm which will help you to actually articulate when somebody asks what kind of blockchain I should use, here is my problem scenario and here is the here is the constraints and here is what I want to achieve.

Then you should be able to at least give him a good informed advise on what might suit his cause the best. But more than that, what you would be able to do is that, once you actually start programming, you know either in Ethereum or Hyperledger or any other blockchain framework forum or whatever, you will actually be aware of all the different paradigms.

So that you can see that whether you have made the right choice in terms of choosing the platform, choosing the architecture of the application, choosing the way you do consensus, etc. So that is what the goal is. And from that point of view, I think this is sufficient information that you need about the IOTA blockchain. So when we come back, we will be talking about the other blockchain that is the Corda.

And that probably is a very different paradigm. Not probably, that is for sure a very different paradigm, quite a bit customized for financial applications. And that should give you a very different taste of a different paradigm. All right, so when we come back, we will see that. Thank you.

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

Lecture - 26

Welcome back to Blockchain Technology and Applications. So last session, we talked about the IOTA and the Tangle blockchain. Now as I mentioned before, the other blockchain that I want to cover is the Corda blockchain, which was created by a company named R3. And it kind of originated in the space of Finance.

So various finance companies, they actually have different types of contracts which are actually written in legal languages. And these contracts actually are such that over time various events, especially time driven events happen, when certain activities are executed based on what is written in the contract.

So as a result, what happens is that over time the contract state changes from just inked contract to maybe some part execution of the contract, and then another part execution of the contract and so on. And eventually, the, these activities may repeat for a while or the contract state might reach maturity.

Whatever happens all this information currently, well before Corda or similar type of solutions came in, came into existence was kept in each individual contracting parties, maybe two banks or one financial institute and a company or individual borrower or lender and a bank, etc. These contracts are kept in each of these entities' individual IT system.

And they might have very different understanding of what the contract really implies or what activities need to take place or even about when an activity has taken place, whether it has taken place properly or whether one entity owes something to the other entity. This has to be reconciled manually, in order for them to come to a shared truth perception.

So in order to make these things more autonomous or and more smooth this notion of a blockchain based solution came in. And the idea was that the existing blockchain solutions such as bitcoin or Ethereum, or even Hyperledger was too generic for this specific application they had in mind.

So therefore, Corda from the very start was very much customized for this kind of financial legal contracts, their execution and keeping each party involved in such contract and contract execution on the same page without having

to do any kind of manual reconciliation. Now, if now financial Institutes they actually enter into similar contracts with multiple different other financial institutes.

And therefore they have many different contracts ongoing at the same time, but not every institute that they interact with are party to each and every contract right? So contracts maybe a two party contract or maybe three party contract, but whatever it is not every other entity for a particular specific contract would be interested to know the terms of the contract, nor do the individual parties involved in that contract for the sake of confidentiality want them to know about it.

So therefore, a completely public ledger or completely shared ledger will not solve the need of this particular type of application. So therefore, the design of Corda was quite different from the blockchains that we have seen before. It is highly customized. It is a permissioned blockchain and it is also, you know, a need to know basic information sharing. And consensus is also limited to the parties that need to consent on any transaction.

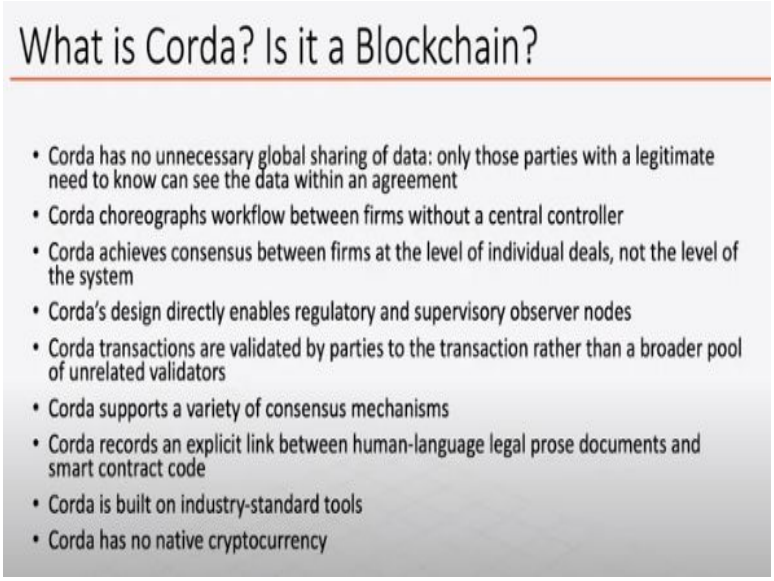
And the transactions in this case would involve execution of a particular activity envisioned in the contract and corresponding change in state of the contract.

(Refer Slide Time: 06:41)

So with that in mind, let us look at Corda. So most of the material in this you can actually download this to whitepapers. One is a very high level whitepaper for Corda and then the other one is the technical whitepaper. And then Richard Brown and Razi Rais they have written a number of blogs, which actually explain this whitepapers and other related technological details in various blogs which also you can find online.

However, we are not going into as much depth about Corda as we have done in the case of Bitcoin, Ethereum etc. But just to give you a flavor of a very customized blockchain for a very specific niche application, we are going to discuss Corda.

(Refer Slide Time: 07:50)



So the first question that one would like to ask is, what exactly is Corda? And is it a blockchain? So there are many differences between Corda and a blockchain like Ethereum or Hyperledger, etc. So the first thing that would jump out of the whitepapers is that Corda has no global sharing of data, because not everybody or every party involved in this data sharing platform would be interested in what is happening between parties, which are engaged in a legal contract regarding some financial activities.

So only those parties with legitimate need to know can see the data within an agreement. Now Corda has another concept called a flow. Basically, when you talk about a transaction, even in case of bitcoin or Ethereum you talk about a transaction which involves different entities and which has a workflow.

So for example in bitcoin if you are making a payment transaction, the workflow would be actually to first consider one of your UTXOs from which you will be used as input to your transaction. Then you have to create two recipients or one recipient or multiple recipients based on how you want to utilize the unused coins that you have that you are possessing by virtue of that UTXO.

And then you will also create the corresponding bitcoin script with the scripting language and then you will broadcast that transaction. Then each node which sees this broadcast transaction will actually execute the bitcoin script by

taking a part of the script from the UTXO transaction and part of the script from this transaction and execute it and if there is a flawless execution, then it will correspondingly consider the transaction to be valid.

But that is not the end of the transaction being permanent or somebody else the one of the recipients to spend the coin that you have given that person, because this has to now make into a block and then all the nodes will make their own blocks and some of them will take this transaction into their block. Some may not take them into their block and then what will happen is that all of them will engage in solving the proof of work puzzle.

And then whoever wins will then broadcast it and then others may also solve the puzzle and they might also broadcast it and then for a while there will be a rest condition. Then whichever part of the if this block in which your transaction has been included, has been there added to the blockchain and then maybe six more blocks have been built on top of it, only then the other party can start using that coin that he or she received from this transaction.

So that is the entire workflow for exchanging or transferring some money from one entity to another in bitcoin. Similarly, in case of Ethereum, you have a workflow that your smart contract you know, in your transaction, you invoke some function in a smart contract and then it goes to everybody and then everybody executes the smart contract, you know, as per required for this transaction.

Then they change the state information and then it may become part of the next block etc. So there is always a transaction is not just a single just you know you just send a transaction broadcast a transaction and money changes hands or our information gets recorded. No it does not happen like that, there is a flow. So Corda uses this.

So flow is some kind of a first class concept in Corda and so the flow is choreographed and this flow is choreographed without a central controller as required by any blockchain platform. Now consensus normally in bitcoin or Ethereum happens by participation of everybody who wants to get involved in the consensus by the proof of work.

And the reason why a proof of work is required is because we do not we cannot trust any single entity that is participating in the blockchain. But we have seen that in Hyperledger, the consensus actually is much simpler, because it is a permission blockchain, everybody has a digital certificate which binds that participant's real world, real world identity to its digital identity.

And therefore, we basically make the consensus very simple by having a set of nodes or a single node even in the case of solo where they actually do the ordering of the transactions. And as far as consensus is concerned, the consensus about the ordering is what Hyperledger cares about and it is done by kind of like offloading that service to the consensus service, which may be implemented as solo as raft which is crash fault tolerant or by some kind of a Byzantine fault tolerant mechanism.

So Hyperledger is the consensus is not at the level of the system, but more at the level of the transactions or individual deals. So same thing in Corda. In Corda the consensus happens between the parties that are involved in that transaction and nobody else, right. So and in Corda also there is a it is a permission blockchain and every participating entity has a strong identity.

And therefore, although one can say that through some mechanism of certification, one can actually make these entities anonymous, but they are, that is not the main purpose of Corda. So Corda is also designed in such a way, so that there may be observer or regulatory nodes, which have visibility to all the transactions, all the activities, and therefore, and we know that banking and finance is a very highly regulated industry.

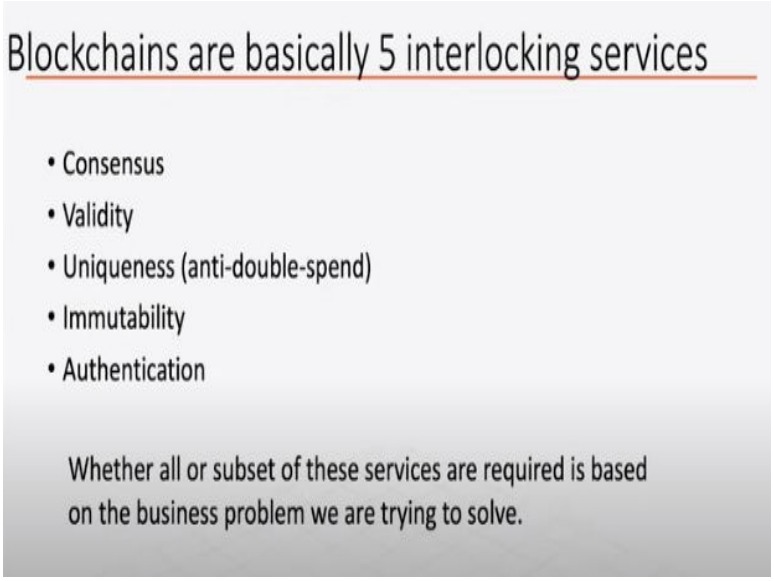
So therefore, regulators need to look at the activities. It has to be transparent to the regulators. It does not have to be transparent to everybody else, but the regulators might want to see that the transactions have no irregularity and so on. So Corda is designed in such a way so there may be observer nodes and regulatory nodes which are looking at all the transactions and also be able to do auditing.

So transactions are validated also by parties to the transaction rather than from any random participants in the blockchain. Corda also like Hyperledger does not make consensus as completely monolithically entangled into the platform. So therefore, Corda also supports a variety of consensus mechanisms. It records obviously, now the most of the finance contracts, they are normally written as human language like English language, legal prose documents.

But when the terms of the contract are to be executed, then that has to be done, you know, using some kind of program, and that is the smart contract. So smart contracts and legal prose documents are kept in link. So we will see that you know in Corda the records there are explicit links between the code that is written in the programming language for smart contract and the legal prose of the contract.

Corda does not create its own programming environment and other things. It is built on industry standard tools, mostly it is based on Java technology. It uses messaging technology, all that stuff is already existent as industry standard. And Corda has no native cryptocurrency because that was not the purpose of creating Corda.

(Refer Slide Time: 19:17)



So the Corda, if you look at their whitepaper, they think of a blockchain as five different interdependent services. Obviously, there has to be a consensus for consistency, there has to be validity for a transaction to be valid. And that concept of validity depends on the semantics of the particular application in hand. For example, in case of bitcoin, the transaction is valid if the signatures match the person who is spending the coins and if the if this is an unspent coin.

In case of Hyperledger, the validity is based on the peers checking that the transactions are valid in the they actually simulate the execution of the transaction and also you know create the result of the transaction as read/write sets and at the end when after the ordering service the nodes before they persist the transaction into the blockchain they actually check about the dates of the transactions.

Of the various pieces of data that is used for a transaction execution and if there is some, you know inconsistency then they also declare a transaction to be invalid. So that is the different notions of validity based on the application semantics. Then there is the shake of uniqueness. And this is very important for checking for double spent. Now you might say that if you do not have a cryptocurrency, then what is doubled spent?

Now if you are having a legal service, right, and a financial legal contract. And let us say you are obliged to pay a certain amount to an entity for you know, at certain time, and let us say you by mistake or for other reason, you execute it twice. And then you are actually making a mistake because you already have, you know fulfilled your

obligation but if you try to do the same obligation twice, it will be kind of like a double spend and you do not want that to happen.

Of course the records have to be immutable because that is the whole point of you know data integrity that one of the major attractions of blockchain. And in the case of most blockchain this immutability is actually guaranteed by cryptographic hash function usage in a clever way in terms of hash chains basically. And then authentication is an important part of the process.

So therefore, you know of course different blockchains have these five interlocking services in different ways. Some of them mix them in the same functionality and some of them have them separated. So in Corda, we have all of these. But, you know as we will see they are used in they are organized in a certain way.

(Refer Slide Time: 23:28)

So according to the originators of the Corda, the question always is, should be that what business problem I am solving by inventing a new blockchain platform or using the blockchain some particular specific blockchain platform. So if you think about bitcoin, the idea is that the business problem we are trying to solve there is that no one should be able to stop me from spending my money, right? So that is the idea of bitcoin.

(Refer Slide Time: 24:12)

Business Problem of Financial Institutions

"The financial industry is pretty much *defined* by the agreements that exist between its firms and these firms share a common problem: the agreement is typically recorded by *both* parties, in *different* systems and **very *large* amounts of cost are caused by the need to fix things when these different systems end up believing different things."**

Imagine we had a system for recording and managing financial agreements that was *shared* across firms, that recorded the agreement consistently and identically, that was visible to the appropriate regulators and which was built on industry-standard tools, with a focus on interoperability and incremental deployment and which didn't leak confidential information to third parties. A system where one firm could look at its set of agreements with a counterpart and know for sure that:

"What I see is what you see and we both know that we see the same thing and we both know that this is what has been reported to the regulator"

So the business problem that and I would leave it to you as an exercise to figure out how this business problem is solved by the bitcoin blockchain. So the business problem of the financial institutions, and this is something that I took from, I believe in from the one of the whitepapers, but this is this prose is quite, quite verbose. But I thought that this would be very important to communicate to you.

So let us read what they have said. The financial industry is pretty much defined by the agreements that exist between its firms. And these firms share a common problem. The common problem is the agreement is typically recorded by both parties in their different IP systems. And in the earlier days, it used to be paper files. And it turns out that oftentimes, this information as recorded by the different parties differ substantially in terms of semantics.

And therefore, a very large amount of cost are caused by the need to fix things when the different systems end up believing different things. So we have different parties, party to the same legal agreement, but they record it in their own systems and when they come to reconcile they if they seem to believe that they are they have a different interpretation or inconsistency then there is a usually most of the time, there will be arbitration, legal challenges and so on so forth.

And there is a lot of cost associated. So the question is, how can we solve this? So imagine we had a system for recording and managing financial agreements that was shared across firms that recorded the agreement consistently and identically. That was visible to the appropriate regulators and which was built on industry standard tools, with a focus on interoperability, and incremental deployment, and which did not leak confidential information to third parties.

So you want a system which will record and manage financial agreements. And this information this recording should be shared across firms. So no longer the firms will have their own IT systems storing this records. They will share the record and the evolution of the record. And this record as they become part of the record, as it become part of the system shared by all the parties, there has to be a consensus before they are put into the system, right. Because everybody is looking at it.

Everybody who is party to this agreement will have to look at it and agree that the interpretation is consistent for all the parties. And so the and this record this will be recorded consistently and identically and it will be visible to regulators as well not only the parties involved but regulators. And it has to be built on industry standard tools that we can trust to be working and functional and error free and effective etc.

And this has to be interoperable between different firms. And it has to be designed in such a way so that the legacy systems are not completely it does not have to be completely replaced by this. So there will be incremental deployment and more important and most importantly, it should not leak confidential information like the contract information between two parties to third parties.

So a system where one firm could look at it a set of agreements with a counterparty and know for sure that what I see is what you see. And we both know that we see the same thing. And we both know that this is what has been reported to the regulators. So all parties see the same thing. They also know that all parties are seeing the same thing and they also know that this has been reported this way to the regulator.

So there will be no inconsistency. Otherwise, the report by two parties will have different things about from the regulator. For example, if you are paying tax, and if your employer sends TDS information to the tax authorities, which does not match the one that you reported in your tax return then it costs the tax authorities to do audit and then you know penalize or whatever. So it basically costs a lot of thing.

So if we could have a shared information system in which tax authority, employer and employee all are looking at the same shared system, and then there cannot be any sense of difference in interpretation, difference in the state of the system, and that would basically make things much more efficient and cheaper.

(Refer Slide Time: 30:46)

So what are the kind of things that the financial institutes agree on? So here are some examples. So Bank A And bank B agree that bank A owes 1 million US dollars to bank B repayable via RTGS on demand. So this is the kind of agreement. So here you see that this agreement requires some action on some event. So the event would be a demand from Bank B and then the action from Bank A should be an RTGS based transfer of $1 million.

So this is the evolution of the contract. Once this is done, this contract can be closed because it is already the obligations in the contract has been discharged. So this is a state evolution of the contract. Similarly, so this is a what is called a cash demand deposit So another example would be bank A and bank B agree that they are parties to a credit default swap with the following characteristics.

So characteristics are not described here. So credit default swap is a very complex financial instrument that banks often do. And I cannot tell you exactly at some point, I used to know what credit default swap means, especially in 2009 when this credit default swap was, had got a really bad name because of the banking crisis that came about in the US and other places in the world in 2008.

But the point here is that this is some kind of a contract and that requires some action at some future time. Another example is that bank A and bank B agree that Bank A is obliged to deliver 1000 units of BigCo common stock to bank B in three days' time in exchange for a cash payment of 150,000 US dollars. This is a delivery versus payment agreement.

So in this agreement, you see that again there is a time bound within which certain action has to be taken and the cash payment has to also be made. So this is how the contract will evolve and once this is done, this also this contract can also be disposed of. So this is these are examples of the legal contracts that these institutions get into.

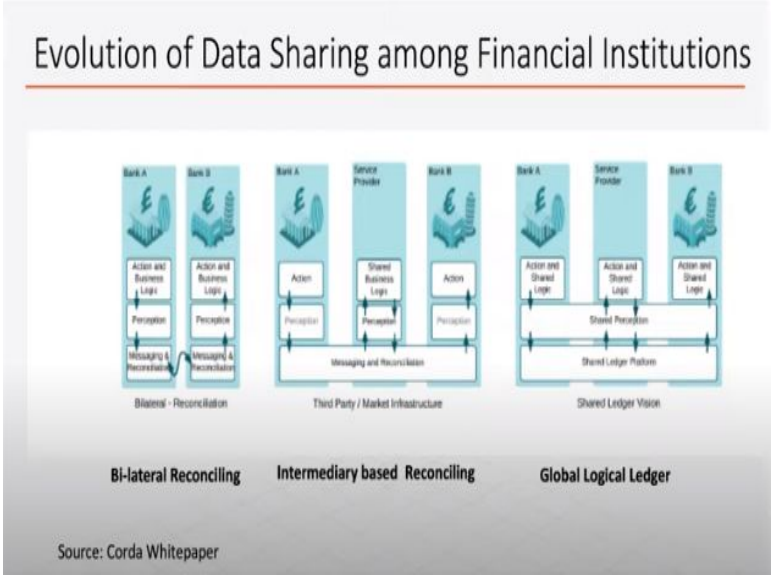
(Refer Slide Time: 33:57)

So we want a system through which all the parties involved in the contract will have this contract and have the corresponding programs or smart contracts that will be triggered based on events based on time bounds, etc. And the contract will evolve in its state. So Corda, as I said before, the difference with other blockchain is that consensus occurs between parties to deal between the parties that are involved in a deal and not between all participants.

Corda lets users write their validation logic in industry standard tools. And they define lets us define who needs to be in agreement on a transactions validity on a contract by contract basis. So a transaction validity is has to be agreed on by the parties involved in the deal and this agreement logic or validation logic has to be can be written in programming languages mostly in case of Corda in Java.

Also Corda also wants to satisfy the CAP theorem that is consistency has to be maintained, availability has to be maintained and partition tolerance has to be maintained. Data is not broadcast to everyone and only to stakeholders who need to know. So this we already discussed that these are some of the differences between Corda and any other blockchain.

(Refer Slide Time: 36:00)



So here is a little bit of history about data sharing among financial institutions. So there is something called bilateral reconciling. So here what is happening let us say we have a bank A and bank B. So Bank A does some action and executes some business logic. Then it has a certain perception about the state of the contract or whatever the deal is.

And then it sends a message to the other bank that I have done this, I have paid $150,000 and my perception is that I should now have 101,000 common stock option for Big company. And bank B at this time has not seen did not have that perception or maybe it has the perception that it has to do it after three days and therefore, it has not done it.

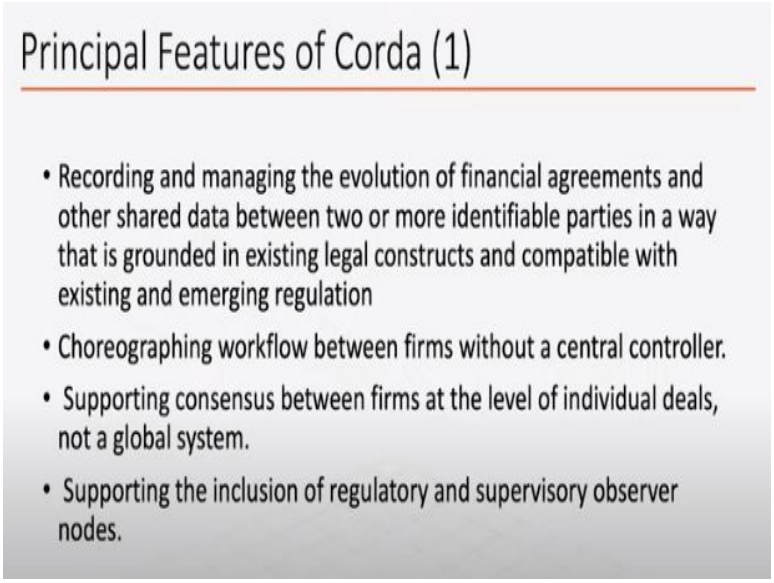
So there has to be a reconciliation or there has to be negotiation that well, if you should, if the bank A was expecting immediately the stock options should be given as soon as it pays rather than waiting for three days. And the bank B thinks that no matter when the fund is paid, it has to be as long as it does it within three days it should be fine. Then there has to be some reconciliation. And that has to be done.

And another interpretation of reconciliation is the reconciliation about their individual ledgers, right. So in my ledger, I have I subtract $150,000. So I should have in my credit 1000 stocks, and this guy should have the opposite and that has to be reconciled. So now, the other possibility is the intermediary based reconciling. So where two parties do not have to directly work with that. So it is kind of like an escrow service.

So the banks basically do the reconciliation through the middleman or a some kind of a intermediary service provider. Now this also has to be done through messaging and reconciliation. Now in the shared ledger vision, the third one where everything is actually there have there may be a service provider, there is bank one bank, A, bank B.

But every time is shared, they put everything on a shared ledger platform rather than having the perception and the ledger individually and then doing a reconciliation letter, they actually have everything in a shared ledger. And therefore, things become much more straightforward.

(Refer Slide Time: 39:22)



So principal features of Corda. So recording and managing the evolution of financial agreements and other shared data between the two or more parties in a way that is grounded in existing legal constructs and compatible with existing and emerging regulation. So you want to record and manage the evolution of the financial agreement. So you have the financial agreement, then evolution means that the obligations are discharged in various stages.

And so the financial agreement is going from one state to the other. And this is done in such a way so that all the parties involved will be able to see this in a shared ledger. And it has to be compliant to any legal or regulatory requirements, legal in the sense whatever is written in the document in the agreement document.

And regulatory means that agreement itself and the discharge of the obligations under the agreement are not violating any regulation. So then, the other thing that it does is that it choreographs the workflow between the firms without a central controller. And that is done through something called a flow which are programmed in usually in Java or Kotlin. And this is something that makes the flow a first class construct.

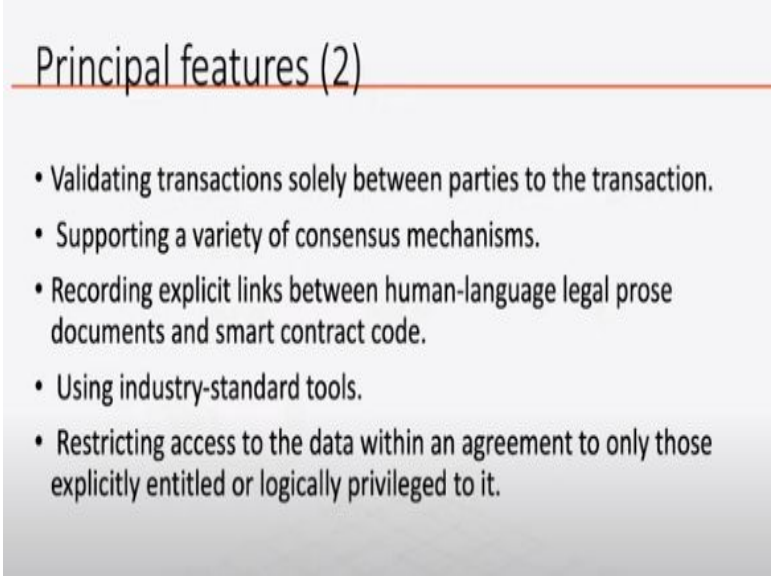
So any transaction has a has to basically invoke a flow that will allow the transaction to complete. And that flow might include the validation, the consensus etc. The consensus actually between firms obviously happens at the level of individual deals and not globally. Obviously, say I am having an agreement with you to exchange something for some money.

Then we need to have a consensus between us about the interpretation or perception of what we are supposed to do and then do it as events trigger us to do it. And that we need to have consensus about that. We cannot, we do not need to invoke anybody else for the consensus. And then we have to support the inclusion of regulatory and supervisory observer nodes. So this is something that is very different.

This can be also done in case of Hyperledger. But Hyperledger does not need this. It in fact, actually Hyperledger can actually do everything that Corda does, but it has to be, you know done by you know molding a particular application for financial agreements and instruments in Hyperledger. And there I can also include a peer which is basically regulatory and supervisory observer.

But it would not be the same as Corda because of certain other things that Corda by virtue of its design provides switch in case of Hyperledger will require a lot more twisting of the Hyperledger concepts.

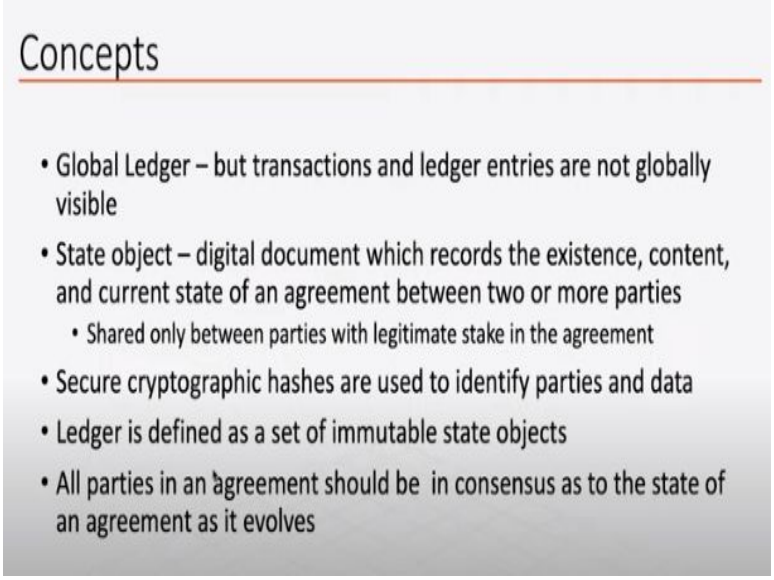
(Refer Slide Time: 43:23)



So also again validation of transaction is also between the parties to the transaction and nobody else needs to do it supporting. So code also supports a variety of consensus mechanisms. You know, it can be solo or it could be some kind of a crash tolerant consensus or it may be some kind of a notary, etc. So recording explicit links between human language legal prose documents and smart contract code is something that is kept in Corda. Use of industry standard tools instead of inventing its own contract, smart contract language etc.

And then it also restricts access to the data within an agreement to only those explicitly entitled or logically privileged to do it. So only the parties involved and maybe the regulatory observer node etc., they can see the data.

(Refer Slide Time: 44:34)



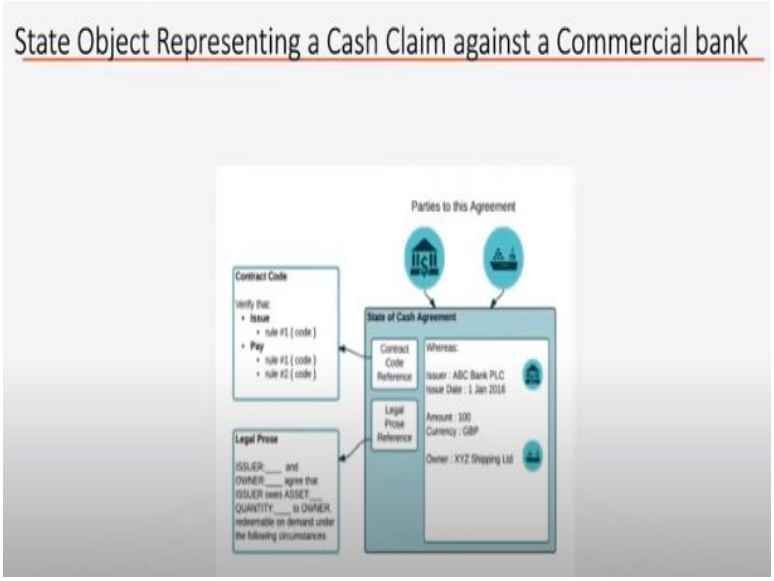
So again, Corda maintains a global ledger. But like the channels in case of Hyperledger, that people who are part of a particular channel can only see whatever happens between those parties who are part of the channel. Here is similar, but not exactly because here in case of a channel, each channel has its own ledger. Here we have a single global ledger, but the visibility of entries in the ledger are not are done in such a way so that only those who need to see those entries can see the entries.

So the what is put into the ledger in case of Corda are called the state objects. So state objects basically are digital document, which records the existence the content and the current state of an agreement between two or more parties. So as I have been mentioning again and again when you have an agreement, that agreement goes through various states based on the execution after different obligations under the agreement.

So those basically are represented as state objects. So in the beginning you are in a particular state, then when some event happens and then you execute something, it goes to a different state and so on and so forth. And this basically going from one state to the other is basically done by doing transactions. So secure cryptographic hashes are used to identify parties and data.

And this is also you know, required for data for integrity reasons. Ledger is defined as a set of immutable state objects. And all parties in an agreement should be in consensus as to the state of an agreement as it evolves, right. So that is the notion of consensus here.

(Refer Slide Time: 46:54)

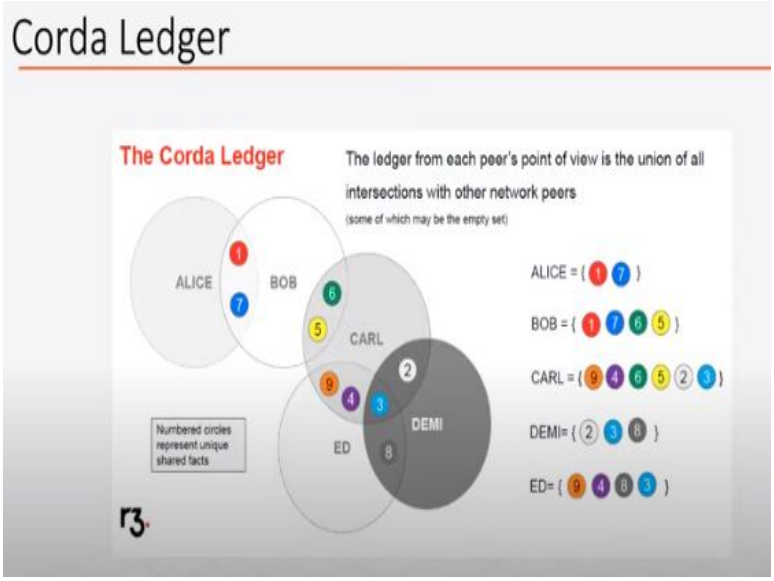


So here is an example of a State object representing a cash claim. Remember cash claim is something like Bank A says that on so and so date I this guy will have to give me this amount of cash, right. So here is an example that there are two parties. Here is maybe a bank and this may be a shipping company. And what it says is that, so it has a contract code, which is basically a programming is a program and here is the legal prose.

So the contract has an issuer and owner. So issuer is the one that issues the cash claim and the owner of the cash claim and the so for example, the shipping company is the issuer in this case. And the owner is the bank. And the issuer has to pay a certain quantity to the owner of the cash claim and then it can be done any time it demands, right. So this is some borrowing mechanism by which this is done.

And this contract has multiple different functions like for example, there will be a contract issue function by which a contract can be issued to a party and also pay function which would have may have multiple different rules, maybe partial payment, maybe total payment and so on. So as we said before the so this is the state object at the beginning and then, as the state changes, the state object will change.

(Refer Slide Time: 48:59)

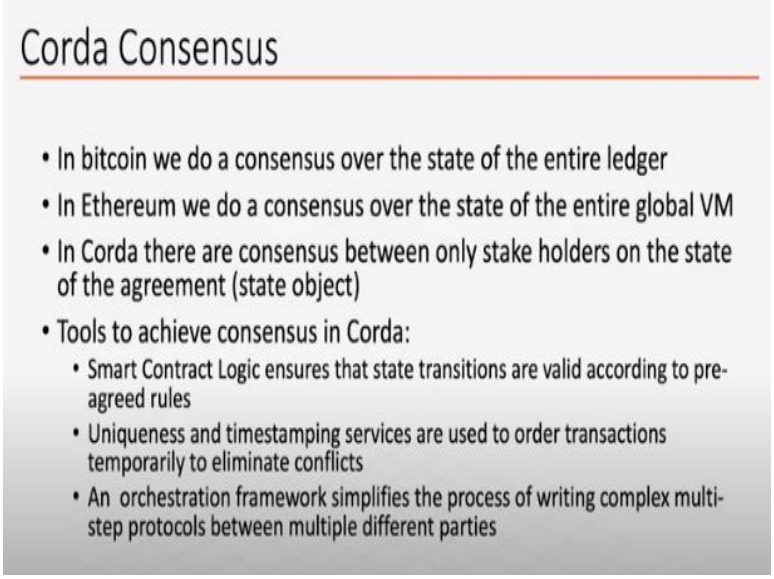


Now here is a pictorial representation of Corda ledger. So here, multiple different this colored small balls, they represent state objects. So Alice for example, has two state objects, basically it is involved in two contracts 1 and 7. Bob is the is also party to 1 and 7 with Alice. But it is also party to 5 and 6 with Carl. But Carl is actually quite active it seems and it has a 2 and 3 that it is involved with Demi and it is also involved with Ed on 3, 4 and 9.

And Ed is involved with Demi on 3 and 8. So 3 is certainly a tripartite agreement. Whereas 4, 9 are bipartite between Ed and Carl. 2 is a bipartite agreement with Demi, but 3 is a tripartite between Demi, Ed, and Carl. And 8 is a bipartite agreement between Ed and Demi. Similarly, Alice and Bob has two different agreements. So this is how the this represents the way the ledger has all these state objects.

So this state object 1 and 7 is only visible to Alice and Bob and maybe a regulatory node. Similarly, 5 and 6 is only visible to Bob and Carl, but nobody else and maybe the regulator node. Whereas 3 is actually visible to all three Carl, Ed and Demi. But 8 is only visible to Ed and Demi. So this is the idea of having a global ledger with very**,** very you know customized visibility.

(Refer Slide Time: 51:08)



So consensus in Corda. In bitcoin we do consensus over the state of the entire ledger. In Ethereum we do consensus over the state of the entire global virtual machine EVM. In Corda the consensus happens between the stakeholders and the state of the agreement. And the tools to achieve this consensus in Corda is, first of all the smart contract logic would ensure that the state transitions are valid according to pre-agreed rules.

Remember, the smart contract is part of the state object. And the two parties have already agreed on that smart contract as well because it agreed to the state object between the two. The uniqueness and timestamping services are used to order transactions temporarily to eliminate conflict. So their uniqueness service and timestamping service.

So timestamping service is invoked to timestamp transactions and uniqueness service we will have some more words on that later. The uniqueness service is the idea that I am going to have the transaction I am going to do consumes some state object and it produces new state objects. But if a state object has already been consumed by another transaction, then I am unnecessarily I am working on stale state object if I am also trying to invoke a transaction with that state object.

And therefore it should not done. So that uniqueness service basically is used to ensure that whenever I invoke a transaction, my input state objects are unique in the sense that it has not occurred in an already happened transaction as inputs. Then the uniqueness is violated, which means that I which basically would mean that I am working on stale data.

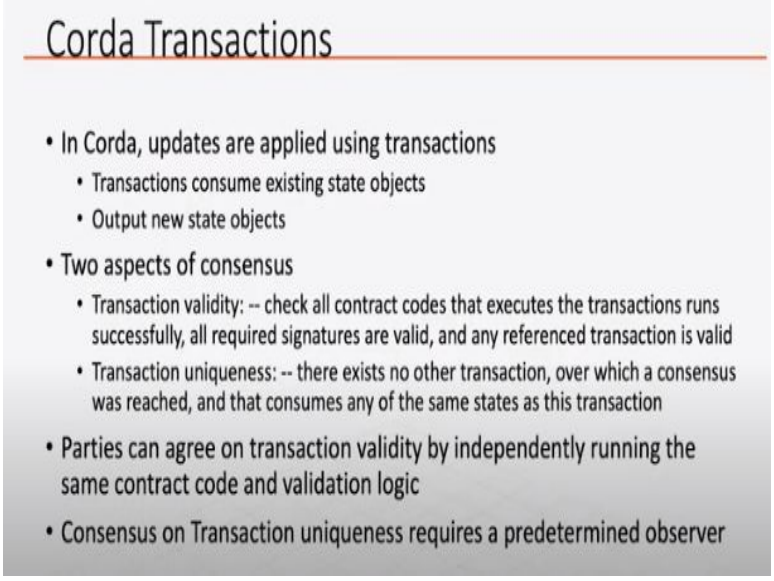
Remember this is the same kind of idea that we have used in Hyperledger when we did the validation by looking at the timestamps on the read/write sets the read set of a transaction and checking against the timestamp or version

number in that case, not exactly a timestamp, version number on the current database that the node has. And that is how it detects whether the transaction is on stale data or on current data and accordingly it invalidates a transaction.

So same kind of thing is done by uniqueness service. An orchestration framework simplifies the process of writing complex multi-step protocol between all this different parties which are involved in executing the smart contract then going to the uniqueness service and timestamping service and come back and complete the transaction. All these activities are part of a flow.

Every transaction needs to have invoke a flow. So there is a flow library for, so commonly occurring transactions can actually use that flow library to orchestrate all the activities and their sequencing needed for a transaction to happen and the state to be properly evolved.

(Refer Slide Time: 54:50)

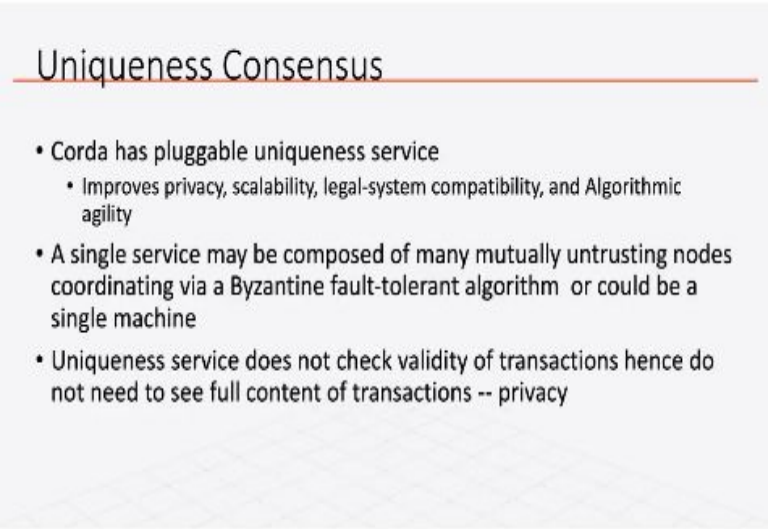


So Corda updates are applied using for the state updates happen to transactions. So transactions consume existing state objects and outputs new state objects. So we already discussed that. So two aspects of the consensus, one is the transaction validity. So I have to check all contract codes that executes the transaction runs successfully, all required signatures are valid and any reference transaction is valid, right.

So that is the validity. So that can be checked by the node that is executing it. But the uniqueness has to be checked through the uniqueness service. So there exist no other transaction over which a consensus was reached and that consumed any of the same states as this transaction. So this is what is called the uniqueness. And this has to be done through a uniqueness service.

So parties can agree on transaction validity by independently running the same contract code and validation logic. But consensus and uniqueness requires a pre-determined observer which records all the transaction what inputs they actually use so that when I go and ask whether the transaction I am going to do my input has been already consumed has been already consumed by another transaction it can tell me.

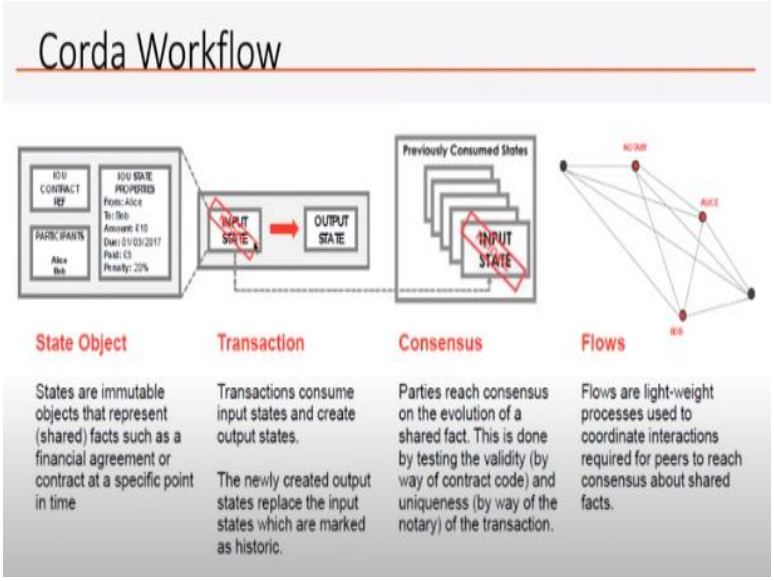
(Refer Slide Time: 56:23)



And that uniqueness service could be actually the has to have an agreement and therefore it can be Byzantine fault tolerant or whatever I feel comfortable with in terms of the availability and fault tolerance etc., or malicious you know assumption about maliciousness node, maliciousness of some of the nodes involving in the uniqueness service. So uniqueness service does not check the validity of the transactions.

So they do not need to see the full content of the transaction, they only need to check the inputs. So therefore there is a some sort of privacy also available.

(Refer Slide Time: 57:12)



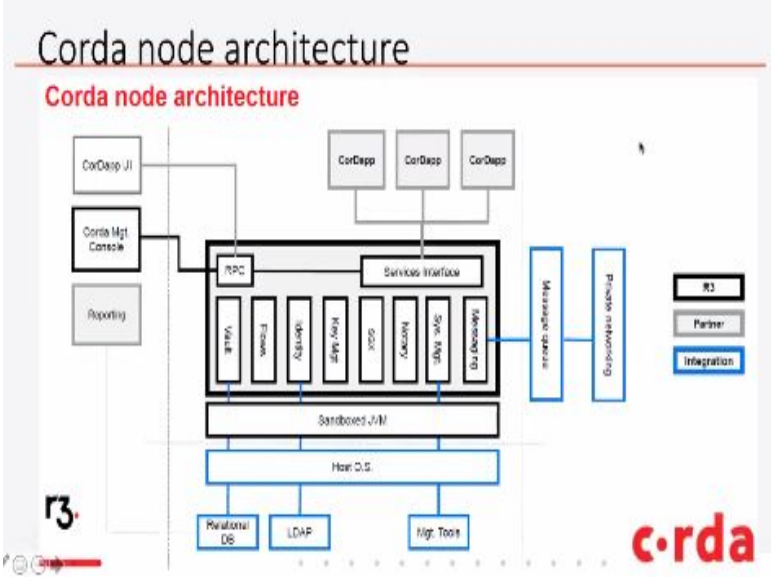
So here is a Corda workflow. So here is a state object. Let us say it is an IOU contract. IOU contract means, let us say Alice, it was issued by Alice to Bob, saying that I owe you $10 10 pounds. And it is due on so and so date and Kevin state is that 5% has been paid, $5 has been paid, and there is a penalty of 20% maybe for paying it halfway or something. I do not know what the penalty is for and Alice and Bob are the participants.

So this is the input state of a transaction. And then a transaction will consume this input state and let us say now Bob wants to pay Alice wants to pay another 5 pounds, but maybe she has to add the 5% 20% penalty. So she has to pay maybe $7 to complete or discharge this contract. So the anyway so whatever the transaction is there will be an output state. And then it goes to the service, uniqueness service.

Uniqueness service has all the previous input states, which are called which are marked historic. So consensus will look that up and tell you that whether it is unique or not, because if Alice has already paid Bob once then invoking a transaction in which Bob is going to get paid again, may not be the right thing. Or even let us say Alice then bought 3 pounds at some point through a transaction.

Now she owes only 2 more pounds. But if I use this as my transaction input then I will think that Alice needs to pay me 5 pounds. So that has to be resolved through this uniqueness service. And that is how this so this entire flow happens like this. So there is Bob Alice, there may be a notary and there may be uniqueness service there may be other services.

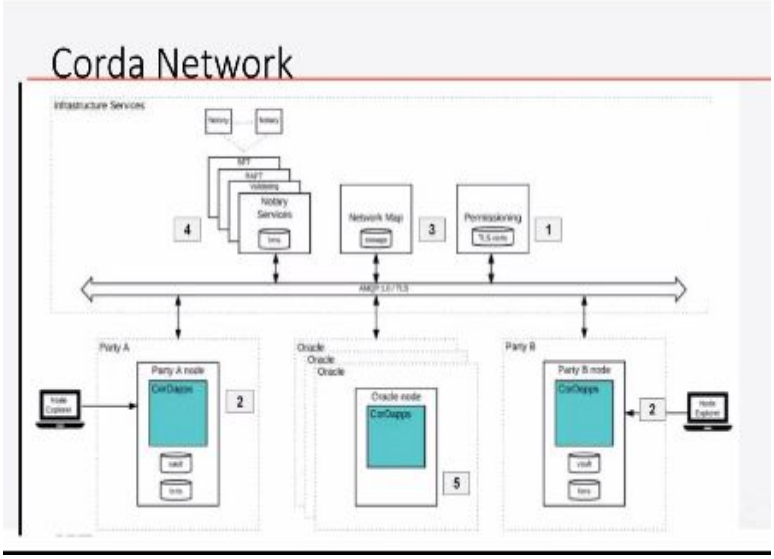
(Refer Slide Time: 59:20)



So Corda node architecture is, so code applications distributed applications written for Corda are called CorDapp. And Corda apps basically run on top of the nodes. And these nodes have various things like vaults. They have the state information, they have flows, they have an identity service, key management service. They have SGX for doing, you know crypto computation.

They have notary and system management functionalities and messaging, right. So they use a messaging service called AQMS. So which is actually a queuing based messaging service. And this there is a sandbox JVM on which all these things execute. And the host operating system may have other things like a database, the name service or some kind of LDAP, etc., etc. So this is the node architecture.

(Refer Slide Time: 1:00:30)



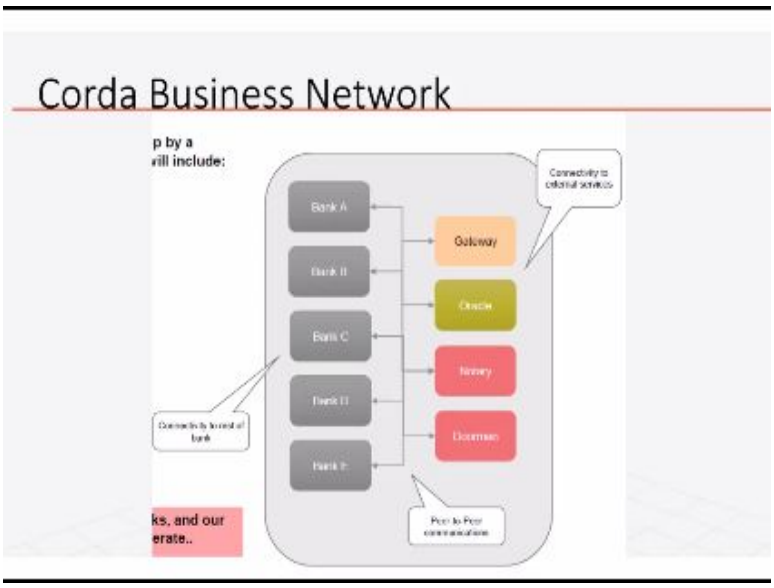
And here is you can see that this let us say you have multiple nodes. So a party A, party B. And then there is notary services, which may be based on BFT or raft or maybe just something. Then we have the network map. So network

map is required for querying how to reach another node when one node wants to interact with another node. And there is a permissioning service which is basically the digital certificate service.

And then there is this message messaging system or AMQP 1.0. And then here are the Corda applications running on the various components inside the node like vaults and other things. And there is another concept called Oracle. In Corda Oracle basically is same as similar as Oracle in Ethereum, where we have seen that Ethereum Oracles basically allows you to store real world information like a stock price etc.

And persist it so that it becomes deterministic to the blockchain and that is the Oracle service.

(Refer Slide Time: 1:02:00)



So that basically is about Corda that I want to talk about. Of course, from what we discussed, you cannot start tomorrow writing Corda applications. But it is not that difficult. So you could actually go to the Corda website and download some already canned examples and start installing and building stuff so it is not very difficult. So we can you can actually try. So this is sort of the Corda business network.

We have various entities in the network with peer to peer communication through that AQMP messaging service. And all these different things like Oracle Notary, Gateway, Doorman, etc., are concepts that are for giving digital certificates, doing Oracle service, doing uniqueness service etc. So this is more or less so but you see that in this case, this is so different from Ethereum or bitcoin blockchain.

This has come very far from bitcoin or Ethereum blockchain. Here the nodes have very different well defined purposes as compared to bitcoin and blockchain where every node is having the same type of responsibilities or same

set of abilities to participate. Of course, computational power might make that little bit curtailed. But at least that is the theoretically every node can do everything.

In the Hyperledger we saw that there are special nodes for peers, there are special nodes for clients, they have special nodes for ordering service etc., but still there is not this kind of custom activities that we are seeing in this one. So this is very customized for financial type of financial, legal financial contract creation, execution and corresponding, you know various say integrity, security etc., for these financial institutions to participate in shared activities and execution of their financial obligations under the agreements etc.

So that completes our seventh week and the next week, the last week we will try to tie up the things that we talked about, talk about some applications and then give you some certain insights if you like, and that is how we will complete the course and we will see you next week.

Blockchain Technology and Applications
Prof. Sandeep K. Shukla
Indian Institute of Technology – Kanpur

Lecture – 27
Summary Blockchains

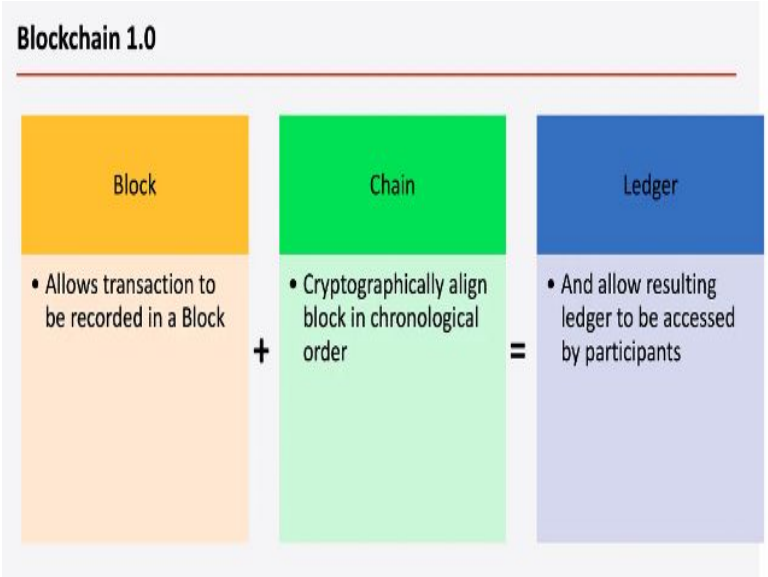
Hello and welcome to everyone for the sessions in the last week of blockchain technology and applications course on NPTEL. So today we are going to have a 3-part week. So in the first part, I will talk about a little bit of summary of all the different blockchain technologies that we talked about and will also mention a few other ones that we did not have time to cover. So then in the second part, we will talk about two applications, particularly in the space of land registry and land record and also in the healthcare information on blockchain.

Then in the third part we will talk about some misconceptions related to blockchain technology and discuss those misconceptions.

(Refer Slide Time: 01:23)

So, I want to kind of summarize the different blockchain technologies going from blockchain 1.0 to blockchain 2.0, all the way to blockchain 3.0.

(Refer Slide Time: 01:38)

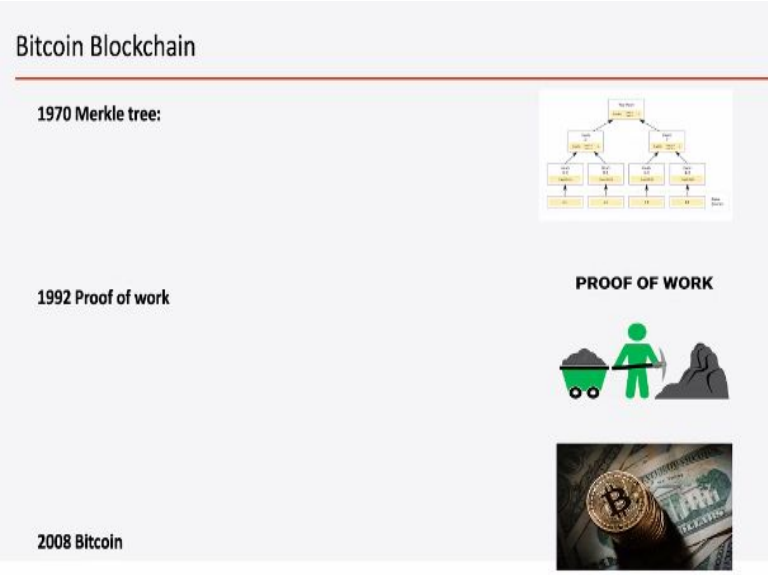


So in blockchain 1.0 which came about with the introduction of bitcoin blockchain back in 2009 and so there the idea is that you would create blocks with transactions and so these transactions are recorded into a block and anybody can create a block and the block that is selected has to solve a cryptographic puzzle as way of proving their proof-of-work, and then once the block is accepted it basically is connected to the latest block before this block by a cryptographic hash function and this is what constructs the chain, right.

So you have blocks and blocks are connected to each other by cryptographic hash and that is what is called a blockchain and this allows you to create a decentralized ledger and this ledger has all the history of all the transactions and this can be looked up by any of the participants in the blockchain you know as new transactions happen and they can be validated by looking up information in the blockchain and also the integrity of the past data is assured because of the hash that is being chained together creating a long history which requires to be completely recalculated.

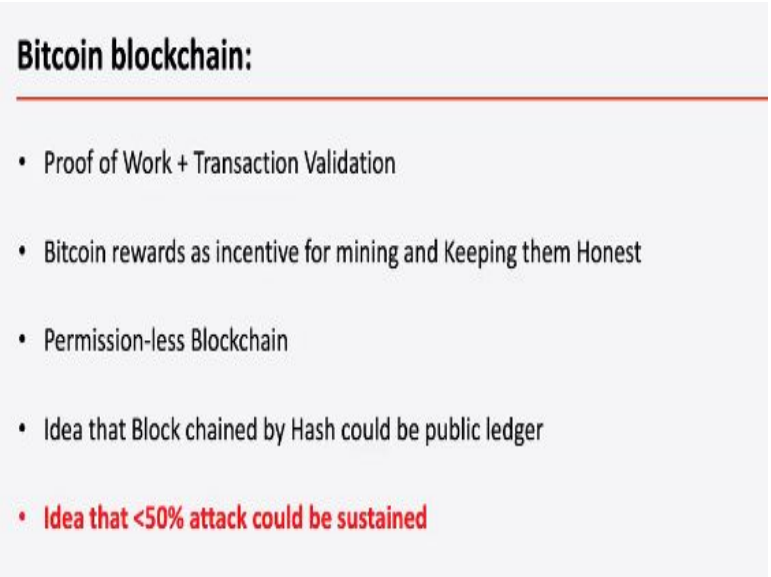
If anywhere in the past you want to make a change making it virtually impossible to change any of the past blocks and that is what gives the temporal resistance property of the blockchain.

(Refer Slide Time: 03:54)



Now Bitcoin blockchain did not come you know out of the vacuum. People have been working on some of the techniques that are used in the Bitcoin blockchain such as creating Merkle tree, the idea of proof-of-work and then this of course the PKI, the public key cryptography infrastructure, use of digital signatures all these things where very cleverly put together into a single technology which is kind of revolutionary and that led to the idea of blockchain in 2008 white paper which eventually 2009 it got published and then the first blockchain implementation of Bitcoin came about in 2009.

(Refer Slide Time: 04:53)



So just to summarize what is there in the Bitcoin blockchain is proof-of-work and transaction validation. This is how the blocks are created and then once the block is distributed throughout the network, everybody has to repeat the transaction validation and then they will put their copy of the block. Once they are satisfied with the transaction validation and the validation that the puzzle was solved by the creator of that block, they actually attached this block in their copy of the blockchain.

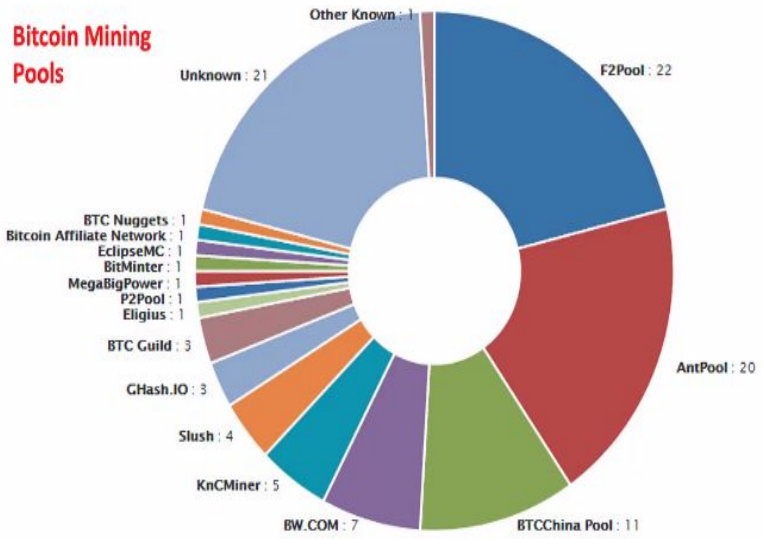
So the blockchain of the ledger is available at many different nodes and that is why it is quite difficult to actually make it tampered with, to make it unavailable and also to make transactions that are not authorized to be done by somebody. Now what is most interesting part of the bitcoin blockchain that we have seen is that it mines these blocks and along with their mining a certain reward amount of bitcoin is given as a transaction in the block to the person who actually or to the node that actually successfully created the ledger block and minors say go on doing this mining.

Not all the blocks are created by the same miner because it is very expensive and there is some amount of stochastic randomization in the process. So a lot of the miners will accumulate this bitcoin, and as they accumulate the bitcoin, their stake in the bitcoin network increases and therefore they will not mine a block that is invalid or that basically is doing a double spending and so on because they will then destroy their own bitcoins.

Because once the value of the bitcoin goes down because of this kind of activities, they are not going to be able to cash the bitcoin that they have won as block rewards and that is basically working against them, so in a game theoretic sense, these incentives keep them honest. Now bitcoin blockchain was purposely built to be permission less and the idea of blockchain by hash is a public ledger obviating the need for banks to keep track of the total amount balance and transactions and so on.

Everybody can have access to the ledger and also it was shown in the white paper that unless more than 50% of the minor of the hash power goes to a specific miner or a group of miner colluding together the tamper resistance and everything in the blockchain will be sustainable.

(Refer Slide Time: 08:51)



Now, we also discussed that the mining is very expensive, computationally very demanding and therefore not any random participant could actually win the process of mining blocks because the amount of computational hardware and power that is required is very high and that is why we have seen that miners pull together to create this blockchain pools, although we did not discuss in detail how this pooling happens.

So, there is quite a bit of technology behind this so that you know when the reward comes how the different pool participants actually distribute the reward among themselves and so on but that you can study, you know there are lot of material out on the internet, you can find that, but what we see here in this picture is that a is a few of the mining pools actually yield a very large amount of hash power. So this thing is always changing, so this is a snapshot from about 2 years ago.

But if you look at the distribution pie chart like this now, it may be slightly altered but may be some of the pools would be renamed or some pools may have come together, some pools might have split, but altogether if you see that about if 3 large pools come together, for example these 3 or I mean 3 in this case may be sufficient, these 3 together will yield a 51%. So 22+20 = 42 and 42+11 = 53. So these 3 pools together yield 53% of hash power which is means that if they collude together, which has not happened, but if they collude together then bitcoin could actually lose its credibility.

So that is one of the big problem with the current situation with bitcoin and we also saw that many of these cryptocurrency blockchains also has this similar situation where they are actually you know getting more and more centralized like we showed you some of our own analysis with respect to Ethereum where we saw that about 100 users out of millions of users actually yield most of the power mostly yield do most of the transactions, they own most of the currency, so this is decentralization that was thought to be not happening with this cryptocurrency seems to be not working so well.

(Refer Slide Time: 12:15)

Problem

- High transaction time- 10 min/block
- Limited size of block
- Wastage of energy
- Pseudo-anonymous

So the other problems with Bitcoin is that the each block takes about 10 minutes and if you look at the average number of transactions that are put in one block, we find that per second about 7 transactions on an average gets confirmed, but even then it is not confirmed because actually you have to build more blocks on top of the block in which this transaction was accepted, at least 6 other blocks before you can be sure that there will be another 4 that would eliminate your transaction.

So in order to be sure that your transaction is actually almost permanent, you have to wait for 10 times 6, about an hour and that makes it very slow for any kind of e-commerce activities, that is why bitcoin has become less like it from a transaction medium and more like an asset-building activity. This block is 2 megabytes and unless there is another 4 this will not change and therefore there is a problem with this by increasing number of transactions per block you cannot really increase the throughput.

But even if you go from 2 megabyte to 4 megabyte, then you will actually only increase the throughput by about you know 100% which would mean that 14 transactions per second type of throughput. Wastage of energy we already talked about that the amount of energy that is used for running the bitcoin blockchain system is the amount of energy many countries as a whole, smaller countries as a whole, use as their energy usage.

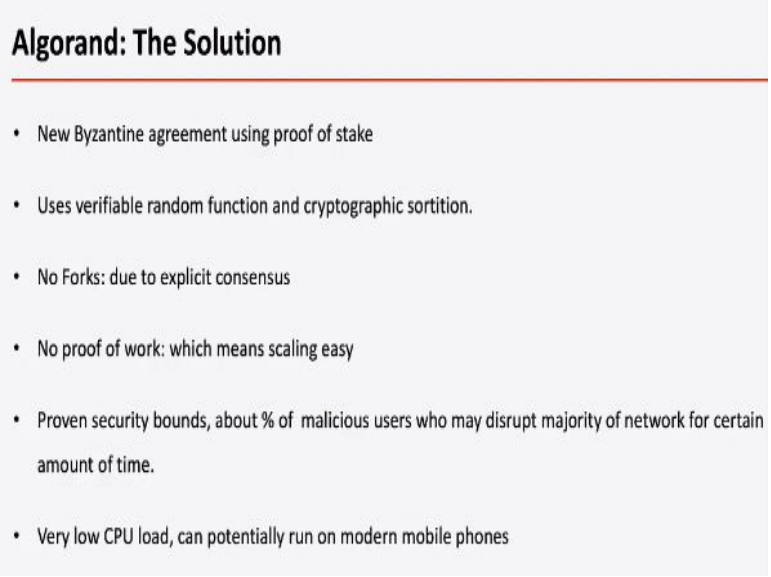
Then the other thing that the original author of the bitcoin blockchain thought about the anonymity turns out this anonymity is sort of pseudo-anonymity which certainly you still cannot you know find the real world identity of a particular miner or a particular participant, but at least you can correlate because all the data is available. So, there are a lot of correlation based way to figure out the activities of a particular bitcoin address even if they actually rename and recreate multiple identities, they also can be correlated. So, all these problems are there in the bitcoin blockchain.

(Refer Slide Time: 15:13)

So, Algorand is another instance of blockchain 1.0 which came about 3 to 4 years ago and here the idea came from Silvio MiCali and his group in MIT and they said that well the bitcoin blockchain uses too much energy for mining and also it's a little too slow because it has to solve the cryptographic puzzle and therefore the transaction throughput is very low and therefore they came up with something based on very sound cryptographic techniques as well as sound distributed algorithms ideas especially an unimproved Byzantine fault tolerance algorithm or BFT.

We did not cover this because it requires a little more involved cryptography to actually go and cover all of this, but I just wanted to show you that something like this exists in the blockchain 1.0.

(Refer Slide Time: 16:39)



The basic idea there is that first of all it uses a new Byzantine agreement and this Byzantine agreement is done by a randomly selected set of participants, not by the entire set of participants, and this is done by some kind of a voting

mechanism. So how do you select random participants from let us say millions of participants that are involved in this blockchain. They uses a very clever idea of something called a verifiable random function and it is actually a combination of cryptography as well as the idea of zero knowledge proof technique.

The idea is that every node runs a function called VRF functions or verifiable random functions that depends on its private key and then based on the result of that computation of this VRF, the result they get will tell them whether at this moment they are part of the committee or not and this process of computing the function also produces if zero knowledge proof, a proof in which you do not give out your private key based on which this function was computed but you sort of provide a cryptographic proof that anybody can check that you indeed computed the random function correctly.

That the result tells that you are right now part of the committee and this committee formation happens in rounds. So every round, so there are multiple rounds are called epochs and during every round, this random function is computed by everybody and then based on the result during that round you will either be part of the committee that votes on the next block or you may be out of the committee in that round. If you have millions of them and if you use only let us say 1000 committee members, then everybody at some point may get a chance to be part of the committee.

However, in order to ensure that you know nobody creates thousands of identities so as to get a higher chance of getting into the committee all the time, so they use this like who gets to selected like this based on how much money you won at that time. So if you have already accumulated certain amount of Algorand currency, then your chance of getting selected is kind of weighted by that and this is kind of use of in an alternative way of using your stake. So if you own more currency in the system, you have more stake in the system.

Therefore you will try not to sabotage the system that is the basic idea of proof of stake in this case. So you get a higher weight age in the process and then if somebody wants to create thousand identities, he has to then distribute the amount in order to be selected, he has to distribute his total stake into these thousand identities, therefore he does not get any extra probability of being part of the committee because your stake has to be now split into smaller stakes.

So that is the idea that they use and then this committee votes in multiple different steps and depending whether the network is at this time synchronous or that it has not been, you know some adversary did not delay stuff in the network out of bound, etc., then it takes less number of steps if the network is being under in some kind of an attack by adversary and the network becomes a synchronous, then it takes a lot more steps to actually converge on the voting.

Once the voting happens, the block that is being voted on whether it should be accepted or not or whether there is an alternative block proposal that should be voted in, either everybody the committee agrees that or at least if all the honest members of the committee, honest in the sense that they are not affected by a Byzantine behavior, if all the honest committees agree on a block, then that block becomes a final a selection for that epoch. In the other case, it may become tentative block and a tentative block will then has to wait until later epochs in which there will be a proper final block selected.

Once that is done, then all the previous tentative blocks get confirmed. So, this is the process and this voting or the or rather agreement is done by a Byzantine agreement algorithm that was not a standard Byzantine agreement algorithm that was and Byzantine agreement algorithm called BA star that was invented for this purpose. So, there is no proof-of-work which means the scaling is much easier, more and more nodes can be added because we are not even making everybody vote, we are always selecting a fix size committee.

Every time I form the committee, I select different set of nodes or some of them may overlap, but you know everybody gets a chance and with higher stake you get a better chance, etc. The idea in the Byzantine guarantee here is that the nodes with two-third of the stake are honest and so at the max one-third of the stake holding nodes would be corrupt or could be adversarial and this is unlike the bitcoin them the mathematical theoretical analysis has been done by the authors of Algorand and there is a pretty good bounds on everything that was done.

Since the process is always done by selecting only thousand nodes and they do not have to do any proof-of-work, this is very low on CPU load and then they can potentially run on modern mobile phones.

(Refer Slide Time: 24:38)

Algorand features

- Sybil attack not possible
- Double spending (safety) not possible
- No Forking (safety)
- Safe from DDoS attacks
- Transaction Efficiency

So, as I said that the way the Sibyl attack is not possible in Algorand is because you weight the nodes by the stake that they have, therefore creating multiple identities but having a fixed amount of money at your disposal does not help you with the sybil attack. So, one more thing is that in Algorand, there is no forking possibilities because everything is done together unlike bitcoin blockchain where everybody is creating their own, anybody who solves a puzzle may create its own block and then they get percolated in the system at different speeds.

There may be temporary forks and then the forks has to join together depending on highest length chain to which people connect their next block, etc. Here, there is no forking and since there is no forking, double spending is not happening because there is always a single chain that is being created. DDoS attack is harder to do in this because normally you do DDoS attack on the nodes that are minors in order to stop the blockchain from growing, now here since you are choosing randomly about let us say 1000 nodes, but they are different from one epoch to the other and this is epochs are very fast.

So, by the time the adversary knows that this is member of a committee, so let us DDoS this one, the next epoch comes and then when the next epoch comes, the committee member is no longer a committee members, so DdoSing that committee member does not stop the process of building the next block from happening and therefore it is kind of safe from DDoS attack. Then the transactions are efficient in the sense that the transaction rates can be much higher because you are doing it in a scalable way.

(Refer Slide Time: 26:55)

Broad Ideas

- Weighted users: Poof of stake
- Consensus by Committee
- Randomly selected committee votes in every round
- Participant replacement: No private state after voting
- Voter cannot be DoSed
- Selection by sortition:
- Users privately find out whether they are members of committee of not
  - Others can verify
- Multi-step Byzantine agreement

So sorry, here is a spelling mistake here, it should be proof of stake. So users are weighted based on their stake. Consensus is done by committee. Randomly selected committee votes in every round and participants get replaced after they participate in the committee. They do not keep any private state after voting, so they have no knowledge that they were part of the committee after that epoch. So therefore, voters cannot be DoSed or subject to denial of service. Selection is done by this process of using VRFs or verifiable random functions, this process is called sortition.

Everybody self selects itself based on the computation of random function and then everybody can verify because during this process of generating this information, there is a proof that he belongs to the committee or not can be verified and there is a multi-step Byzantine agreement called BA star that was invented for this purpose.

(Refer Slide Time: 28:06)

Algorand Advantage

- Security
  - Verifiable Random function protects the committee members
- Speed: Byzantine Fault tolerant algorithm increases speed
- No forks: network never have divergent views of confirmed transaction, even in case of network partition
- Latency: confirm transaction in order of minutes (no need to wait after block accepted)
- Problem: Patented, not open source

The advantage is that it has a lot more security from DDoS kind of attack, from Sibyl attacked and changing the

refreshing the committee every epoch makes it very hard to target more than one-third stakeholders to become corrupt, etc., etc. The speed this Byzantine fault tolerant algorithm increases the speed, you know the mining process and bitcoin is much more time consuming and computationally intensive. There is no fork, so therefore it is difficult to do, even network partition can be tolerated in this because of the way this happens.

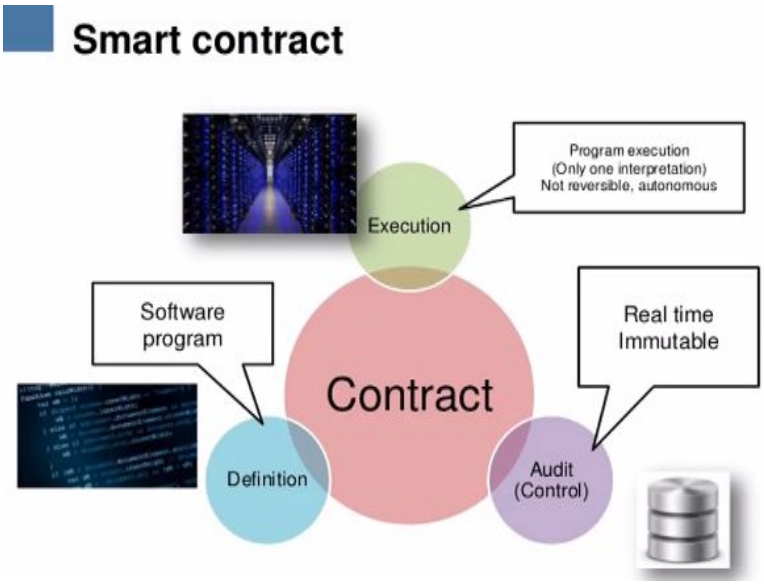
The latency to confirm transaction is in the order of minutes, so you can actually confirm blocks in the order of minutes and you do not have to wait for multiple blocks to build on top of your block before you feel confirmed. So everything gets done very fast. The problem is that it is patented and it is not open source. So therefore, we cannot really you know use it like Ethereum or Hyperledger, etc.

(Refer Slide Time: 30:08)

So now, then we talked about blockchain 2.0 where the idea of the block and the connecting blocks with hash to create a blockchain and then having them replicated all over the places all this nice ideas of blockchain 1.0 is added with the idea of the smart contracts. The smart contracts as we have discussed earlier is basically a computer equivalent of having a contract to do something and that that may also lead to exchange of money or exchange of goods versus money or exchange of labor versus money, etc.

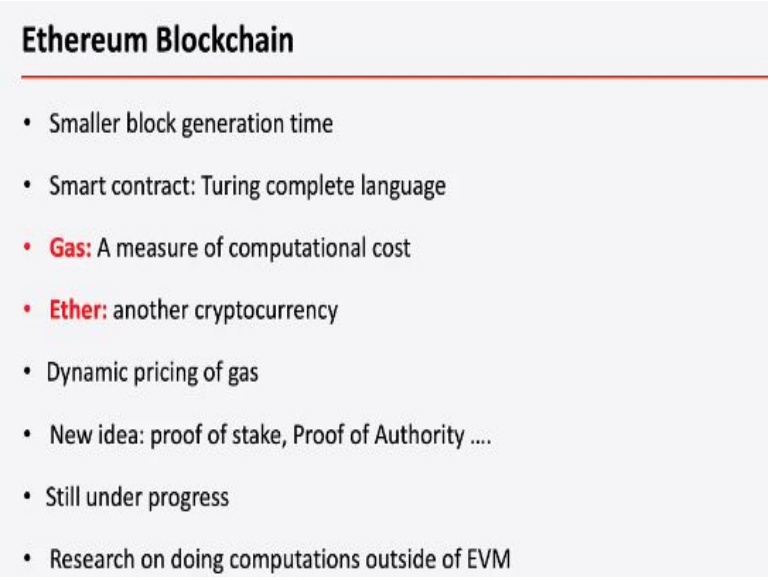
The execution of the contract is often done under the purview of the law. So that is what this picture kind of shows and then an audit trail is created by keeping the contract information when it was created, when it is executed, etc., etc., so then the auditor can check them.

(Refer Slide Time: 31:01)



So, this can be replaced in a smart contract based scenario where all these things are done by computer program. So execution is done through program execution and program execution is done on every node that wants to validate the transactions on the Ethereum virtual machine. The contract itself is a software program or called smart contracts and then the ledger basically allows you to do the audit, so that is the idea of smart contract that basically made it Ethereum the forerunner of the idea of blockchain 2.0.

(Refer Slide Time: 31:41)

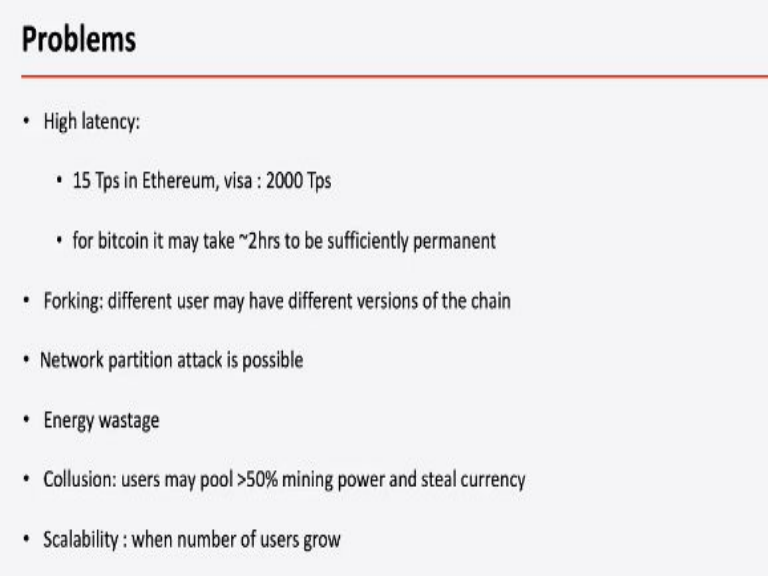


Ethereum blockchain has much smaller block generation time, although it uses a proof-of-work, but it is faster because everything is done pretty autonomously. Smart contracts, Ethereum made sure that the smart contract is done to a Turing complete language which leads to the question of termination going into infinite loop and so on and therefore they introduced the notion of gas. So gas is a measure of computational cost and then if you allocate a certain amount of gas.

Then your program even if it is written in a way that it could have the possibility of going into infinite loop because of the running out of gas that cannot happen. Then it basically has its a native cryptocurrency called Ether and the gas is dynamically priced and you can actually give more price for your transaction to go through for gas and stuff like that. Ethereum now has been for a while is toying with the idea of proof of stake, proof of authority and there are certain implementations already but Ethereum main net is still using proof-of-work.

So these things are still under progress and then the other issue is that EVM is obviously you know it implements execution engine for a Turing complete language, but it is still going to be much slower. So the question is can you do more intensive computation outside the EVM, these kind of ideas are floating around.

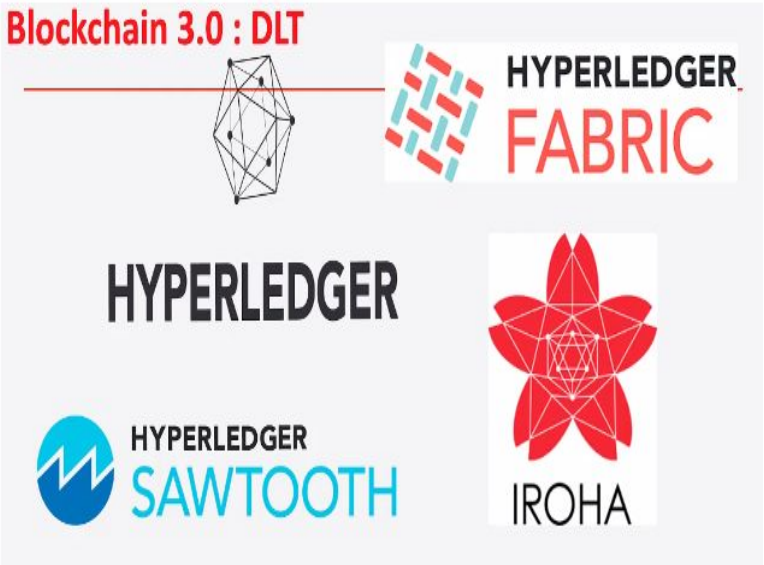
(Refer Slide Time: 33:33)



So it is still of high latency, it is about 15 transactions per second in Ethereum compared to 2,000 transactions per seconds by say Visa, MasterCard, etc. So therefore, there is still a long way to go for it to be competitive against standard ecommerce you know transaction choice for. Of course, Bitcoin is even much slower, so therefore Ethereum is slightly in a better position. Forking can happen. Network partition is possible which can basically create problem in terms of creating forks in the system or double spending to happen which needs to be eventually invalidated.

Energy is still a concern because you are still doing proof-of-work of course in the newer ideas of proof of stake and proof of authority, etc., but reduce that. There is still a 51% attack possible and as the number of users grow, these throughput and everything could actually become much more problematic.

(Refer Slide Time: 35:11)



So the blockchain 3.0 came about in order to introduce, first of all it keeps the idea of the block in the chain, hash chain, the idea of you know double spending proof technique for accepting new blocks. It also keeps the idea of smart contracts all that stuff, but it actually first of all was designed to be permissioned block chain and also the idea of not only permission but also private blockchain because in order to use blockchain for enterprise, finance, or any kind of e-governance application, you cannot have all the data that everybody in the world can see.

So you may need transparency to certain extent, but auditability to some extent, but you do not want each and every data item to be free for all to download and analyze, etc. You also do not want to spend enormous amount of energy to actually generate your blocks and may ensure that your blocks have integrity. You also do not want that any random person can actually take part in transacting or creating blocks for your enterprise information system and therefore this idea of this distributed ledger technology.

So DLT or distributed ledger technology is often you know distinguished from blockchain even though it is kind of like an advanced version of blockchain and Hyperledger is one such example.

(Refer Slide Time: 37:13)

Hyperledger Fabric

- Started from IBM
- Currently owned by: The Linux Foundation
- Open Source-Apache License
- Flexibility
- No mining

So, it started from IBM, but currently it is a consortium called the Linux and it is owned by the Linux foundation and it uses an open source licensing model. It is flexible and there is no mining. It uses for the ordering service which allows you to actually choose based on your need whether you want to be a single node or solo ordering or you want to be a crash fault tolerant ordering like you know Raft type of ordering or you want a Byzantine fault tolerance model ordering. So, ordering processes where a consensus is needed and that is what a Hyperledger does.

(Refer Slide Time: 38:02)

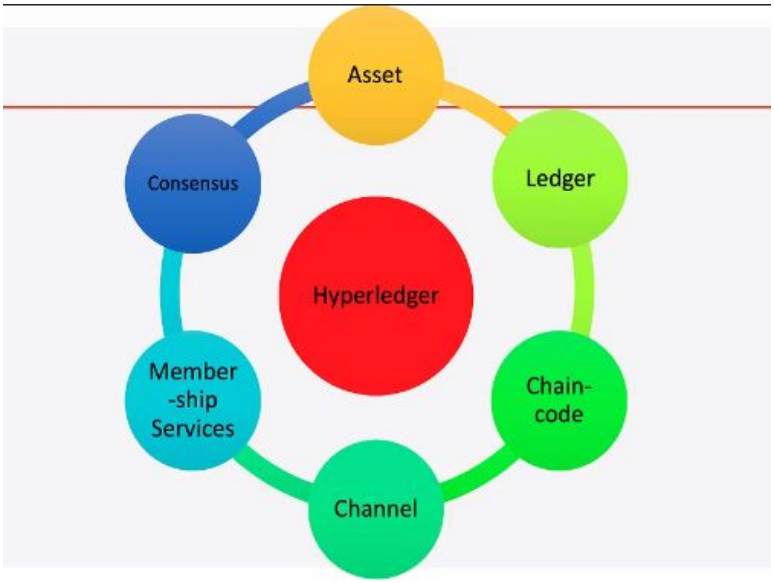
Characteristic	Ethereum	Hyperledger Fabric
Description of platform	– Generic blockchain platform	– Modular blockchain platform
Governance	– Ethereum developers	– Linux Foundation
Mode of operation	– Permissionless, public or private[4]	– Permissioned, private
Consensus	– Mining based on proof-of-work (PoW) – Ledger level	– Broad understanding of consensus that allows multiple approaches – Transaction level
Smart contracts	– Smart contract code (e.g., Solidity)	– Smart contract code (e.g., Go, Java)
Currency	– Ether – Tokens via smart contract	– None – Currency and tokens via chaincode

So, this slide basically shows you the various differences in the Ethereum and Hyperledger fabric because this question comes up often that whether to, because Ethereum also allows you to create a private blockchain that you can run as an Ethereum a network administrator which will be different from the Ethereum main net, but you can do that. So therefore often this question is asked like you know whether to go for Hyperledger type of BLT or Ethereum. So, this kind of comparison often comes about.

So obviously, Ethereum is more generic whereas the Hyperledger is very targeted for enterprise systems. Ethereum is governed by the Ethereum developers, the Hyperledger is governed by Linux Foundation. Ethereum is permissionless, it could be public or private whereas in Hyperledger you are always permissioned and private. Consensus is based on proof-of-work or in the process currently it could be also with proof of authority or proof of stake and the consensus is at the entire ledger level whereas in Hyperledger the consensus first of all is flexible.

You can plug in your consensus and then it is actually at the level of transactions. The smart contracts are in available in both, but hyper ledger allows you to go to go write this code in any generic language that you want. It is basically a piece of code that needs to actually execute some business logic. There is a native currency in Ethereum and then you can also create tokens on top of them. Hyperledger has no native currency and the currency and token can be developed based on smart contracts or the chaincode.

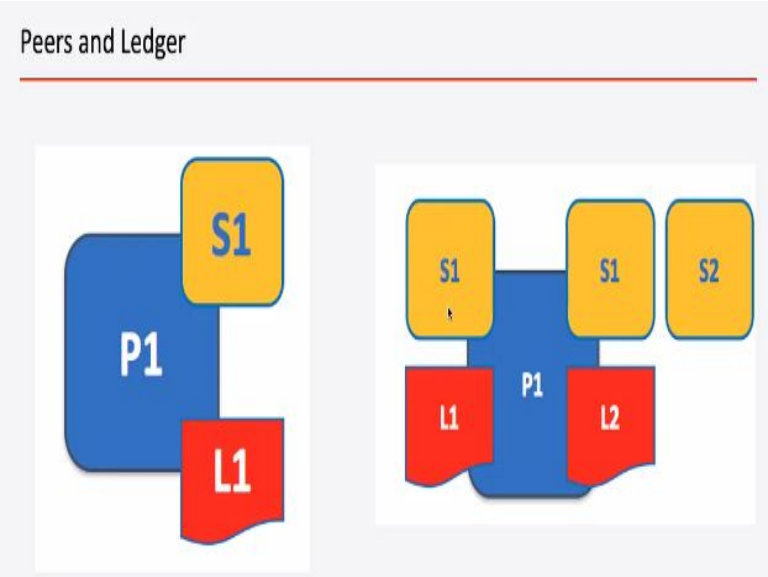
(Refer Slide Time: 40:22)



So Hyperledger has all this different concepts in them. One of the important concept of the membership service that basically gives identity to people and it is also pluggable. You could use a standard digital certificate technique or some kind of even enterprise you know authentication and access control mechanism. Hyperledger also is known to have smart contracts which are called chaincode. Hyperledger also has the concept of channels you can run on the same Hyperledger network multiple private channels between different nodes which will have their own ledgers, right.

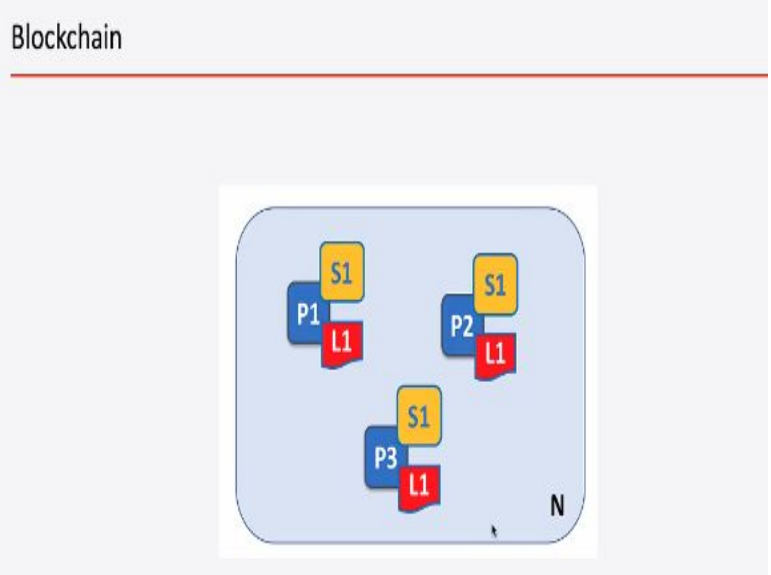
So, it can actually run in such a way so that you have multiple ledgers and you can actually query across ledger, so you can actually create a much more complex enterprise information system. There is this notion of digital assets, basically these are the ones on which transactions happen and then consensus is basically the ordering that happens.

(Refer Slide Time: 41:38)



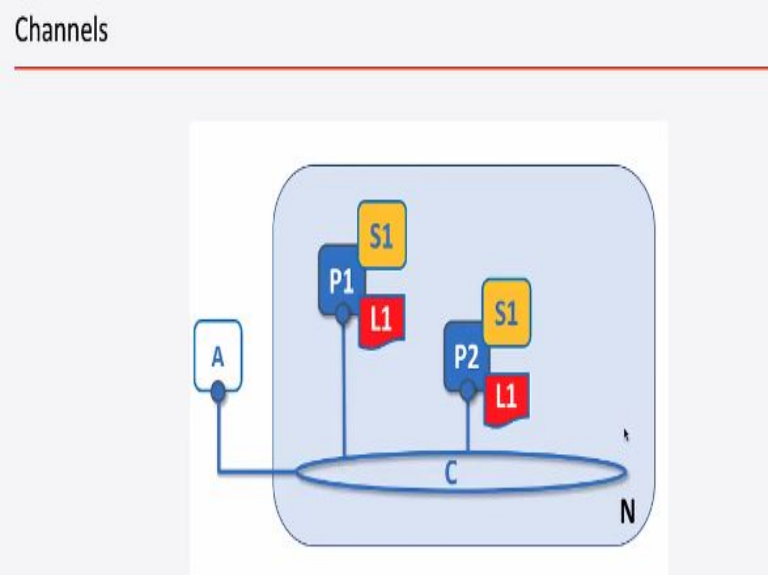
So, this is just a few pictorial representation from IBM's own slides. So every peer will have a docker running in them and the smart contracts run in their own dockers and then they have a copy of the ledger and the world state database and then you can have multiple smart contracts and multiple ledgers. So this peer in this case is participating in two channels, in one channel the ledger is called L1, the other channel ledger is called as L2 and within the namespace of this channel you have 2 smart contracts running on two docker containers and in this channel it runs a single smart contract in a single docker container.

(Refer Slide Time: 42:34)



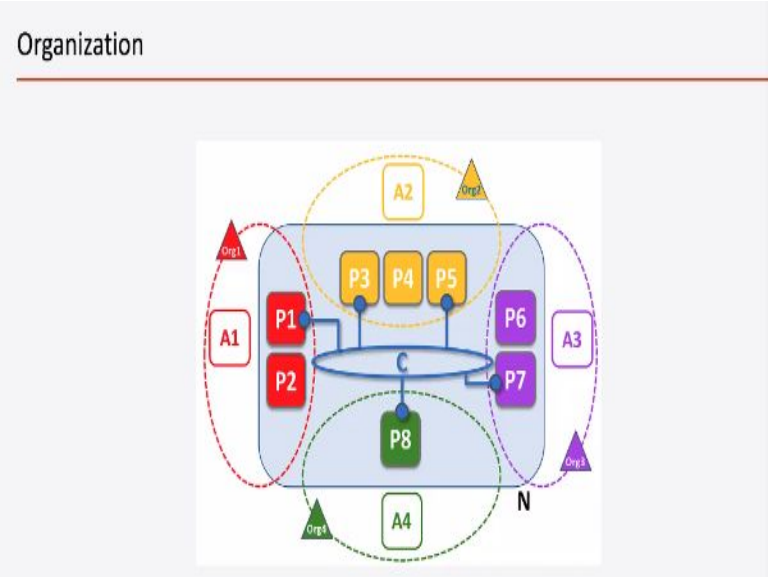
Then the entire blockchain looks like this. Every node in this particular chain or the network, we see 3 peers. They run smart contracts, one one smart contract each and they are on the same channel that is why they are all sharing the same ledger.

(Refer Slide Time: 42:53)



Whereas in case of a channel business, the same peer may participate in multiple channels like in this picture. The same peer is in 2 channels more, in one channel it sees this ledger, the other channel it sees this ledger. The same idea is in this channel.

(Refer Slide Time: 43:13)



So here, we have a single channel with participants, peers they all have the same ledger, but the Hyperledger also has these notion of an organization. So you can have multiple organizations like if multiple banks are using a Hyperledger based distributed decentralized ledger on which they do interbank transaction, etc. Then some peers will belong to this organization, let us say bank 1, this would be another, this may be auditor, this may be another bank, this may be the central bank whatever, right.

So, there are multiple organizations and in the trust idea is that within your organization all the peers you trust whereas across organizations you do not trust the peers. So, there is this distinction of the participants not only in terms of, you know when we studied Hyperledger, we saw that there are nodes that are called peers, there are nodes that are called validators, there are nodes that are called orderers, etc. So already there is a distinction of the type of nodes as compared to bitcoin and Ethereum where every node is at the same footing.

Then in this case also, we are seeing that there is organizational belonging of the peers and then the trust levels are also different.

(Refer Slide Time: 44:59)

Endorsement policy

Describes condition by which a transaction can be endorsed

- A transaction can only be considered valid if it has been endorsed according to its policy
- Each Chaincode is associated with an Endorsement policy

Examples of policies

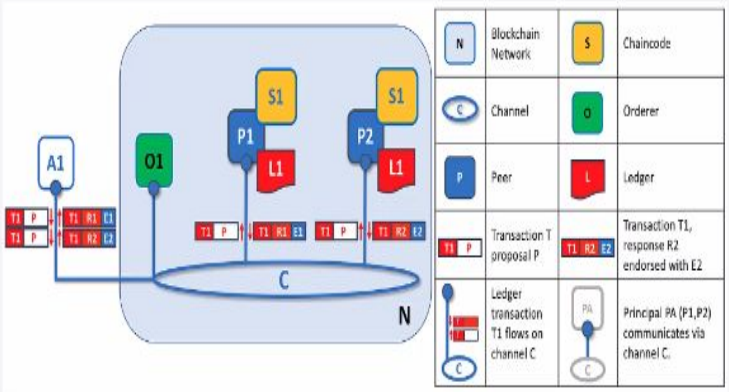
- AND('org1.member','Org2.member','Org3.member')
- Or('Org1.member','Org2.member')
- Or('Org1.member',AND('Org2.member','Org3.member'))

Then we know that Ethereum uses this idea of endorsement. So when a transaction is initiated by a client, then it sends it to certain number of peers which are in the endorsement policy and this endorsement policies might say something like organization ones organization 2 and organization 3 should have one-one peers which would actually all of them must endorse. In this case, we are seeing an Or endorsement where organization 1 and organization 2, either of them can endorse this.

Then we are seeing the idea of and more complex endorsement policy where we have either an organization 1 member or you have to have both the organization 2 and 3 members.

(Refer Slide Time: 46:02)

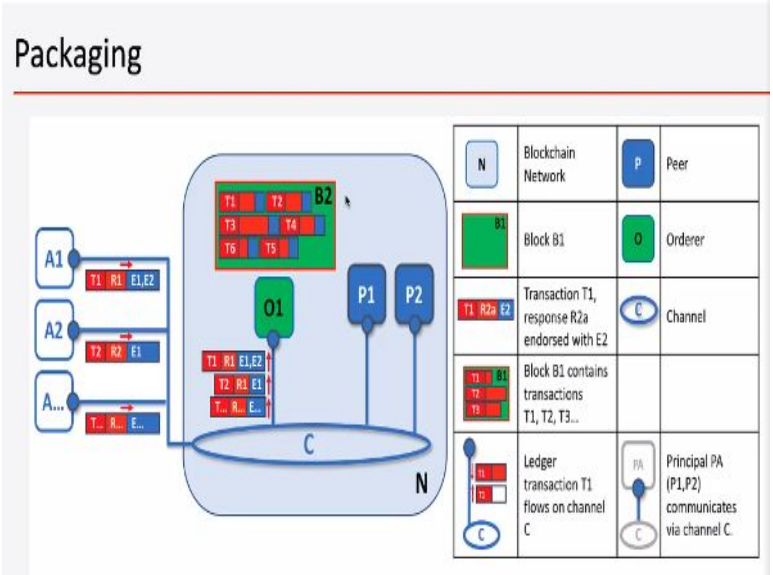
Proposal



So, a proposal looks like this. So a proposal will have the transaction proposed by a client. Ten transaction proposal goes to the peers who will execute the smart contracts, creates the read write sets and then send them and with their endorsement and digital signature, they will send it back to the client and then the client will then send it back for

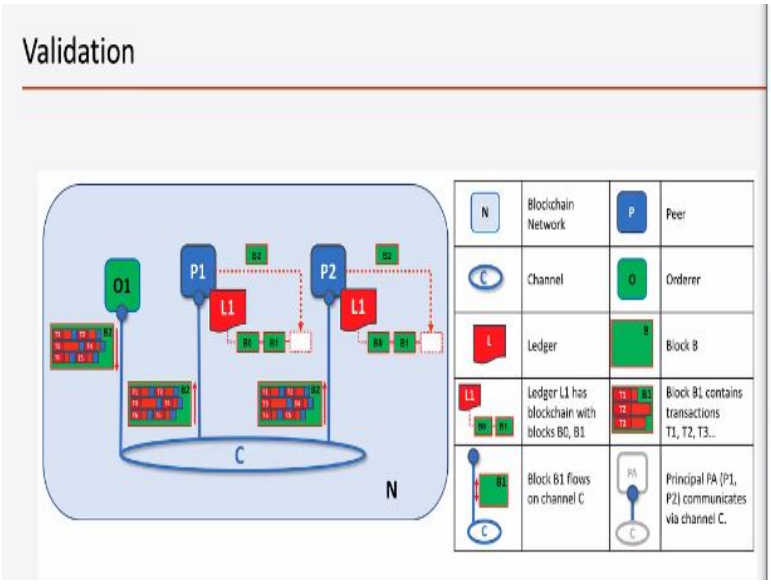
ordering. So that is the idea of a proposal and the proposal being endorsed and then the endorsed transaction collected by the client and then sent back to the ordering service.

(Refer Slide Time: 46:47)



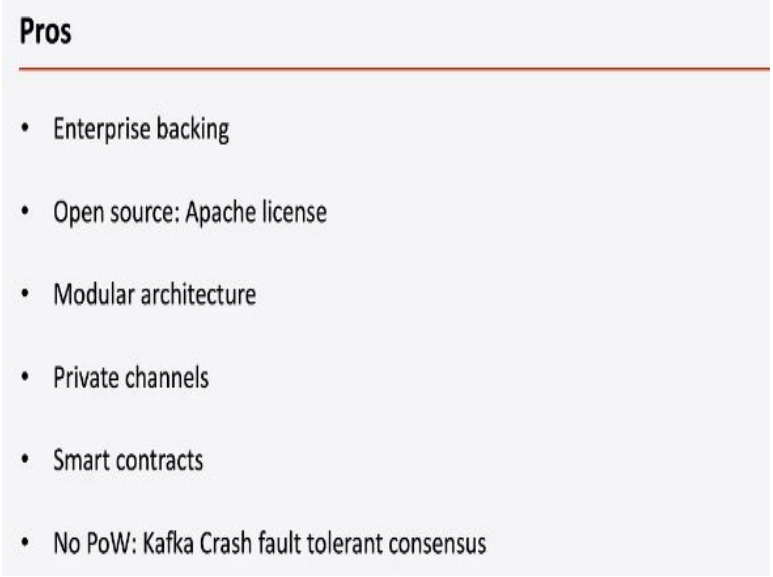
Then once there is sufficient number of these transactions, the orderer will do whatever is required to first of all reach a consensus on the ordering of the transactions across the different transactions, then it will create a block. So that is what the block is, B2 is a block that is being packaged here.

(Refer Slide Time: 47:16)



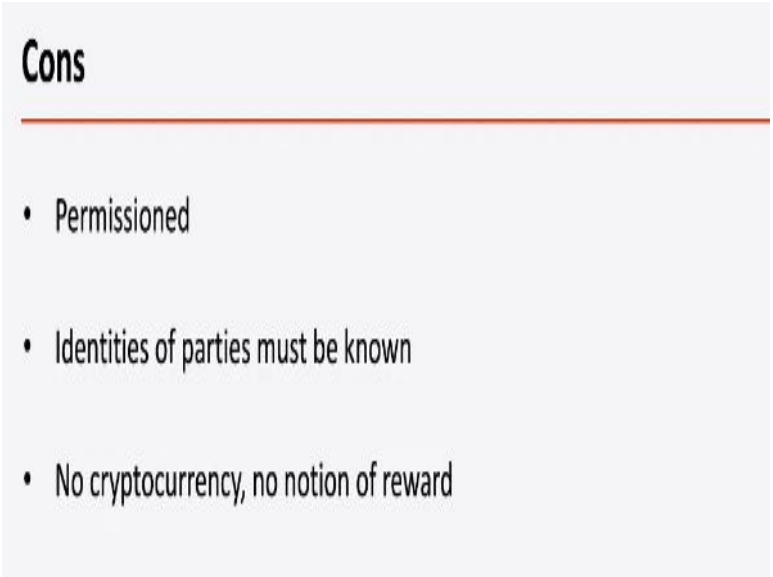
Then finally, every node that wants to validate, they will validate the block and then if they find that there is a mismatch between the read write set versions, etc., they will invalidate the transaction, but this invalidation will be marked on the block, but it will not be actually, you know it is not like the invalid transactions will be thrown away, they will still be in the block but they will make marked as invalid and then every peer will now update their database, take a world state database as well as their ledger.

(Refer Slide Time: 48:04)



So, the pros of Hyperledger are that it has enterprise backing. It has many application examples. If you go to the Hyperledger dot org, you will find a lot of real-life applications. It is an open source. Modular architecture. It has an idea of private channels which helps you to keep certain information flow from another set of information flow and information systems. It allows smart contract and there is no proof-of-work, so you can have multiple different types of consensus as your situation demands.

(Refer Slide Time: 48:48)

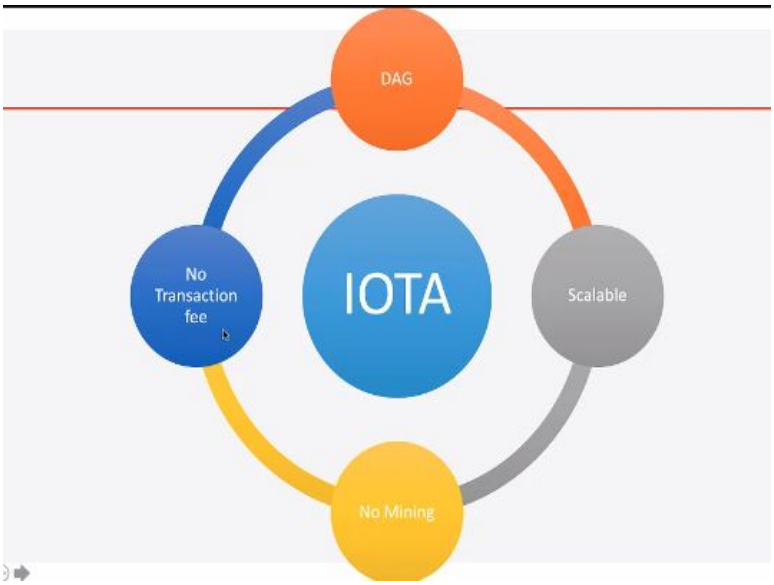


The cons are that it is permissioned. So you have to actually get you know some identity, so identities of the parties are known. Of course that may be considered as cons or may be as a pro depending on whether you know what your needs are in terms of trust. There is no cryptocurrency and there is no notion of reward, which I also do not see as a con because that is not what it was designed for.

(Refer Slide Time: 49:22)

So, then we also showed you blockchain without block and without chain which was the example was IOTA.

(Refer Slide Time: 49:32)



The idea was that there it is not really a chain of blocks, it is actually a DAG, directed acyclic graph, of transactions. It is highly scalable. There is no real mining involved and there is no transaction fee.

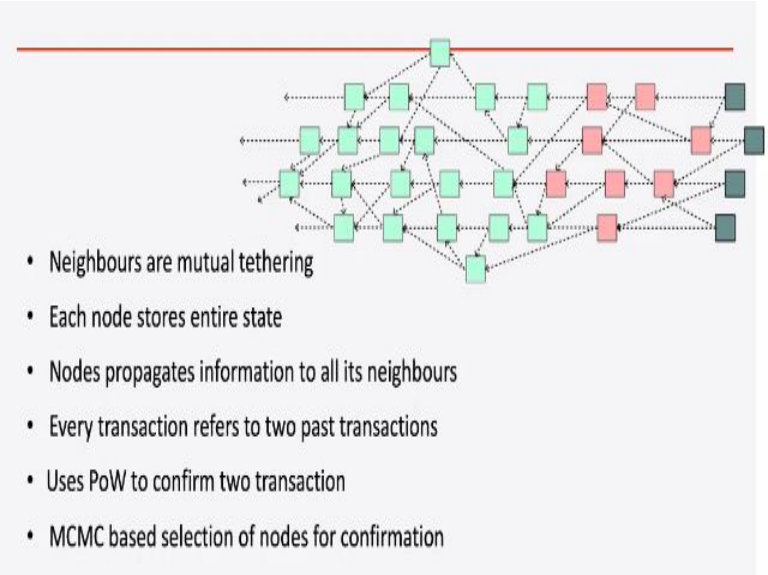
(Refer Slide Time: 49:49)

IOTA Blockchain

- Public Blockchain
- Genesis transaction creates maximum number of IOTA tokenS that could be ever circulated
- Transaction are called sites, unconfirmed one are called tips
- Sites are weighted in proportion to computational work needed
- Does not support smart contract

So, it is a public blockchain. The genesis transaction creates some maximum number of IOTA tokens that could be circulated and never ever a new IOTA in that particular IOTA instance will be created. Transactions are called sites, unconfirmed transactions are called the tips. We saw the tip selection algorithms and sites are weighted in proportion to the computational work needed and therefore we saw this idea of the confirmation confidence and it does not support any smart contract.

(Refer Slide Time: 50:28)



So, the neighbors are basically transactions which are tethered to each other. Each node stores the entire state. Nodes propagate information to all its neighbors. Every transaction refers to two past transactions and it uses a little proof-of-work to confirm the transactions in order to avoid spamming and a Markov chain based selection of nodes is used for choosing the tips. Now we discussed that a lot in the context of IOTA before.

(Refer Slide Time: 51:08)

Pros

- Good for micro payment
- Quantum immune
- No hierarchy of nodes
- Public Blockchain without rewards
- Network becomes faster as it grows
- Offline transaction is possible
- Parallel vs Sequential consensus

Pros is that it is good for micro payment. It has its own invented cryptography which is claimed to be quantum immune, which means that it does not depend on computational problems that are solvable fast in a quantum computer, you know theoretically possible to solve fast on a quantum computer. There is no hierarchy of nodes. Every node is basically a transaction. It is a public blockchain without any reward. Network becomes faster as it grows. Offline transactions are possible, we showed that when we discussed this.

You can have a lot of parallel activities going on at the same time because you can select different tips, different new transactions, select different tips and or same tip but they can all work in parallel versus in the bitcoin on Ethereum, etc. Even in Hyperledger, the consensus is sequential when the block is being made. No other new block can be made in parallel because there is this sequential consensus process.

(Refer Slide Time: 52:32)

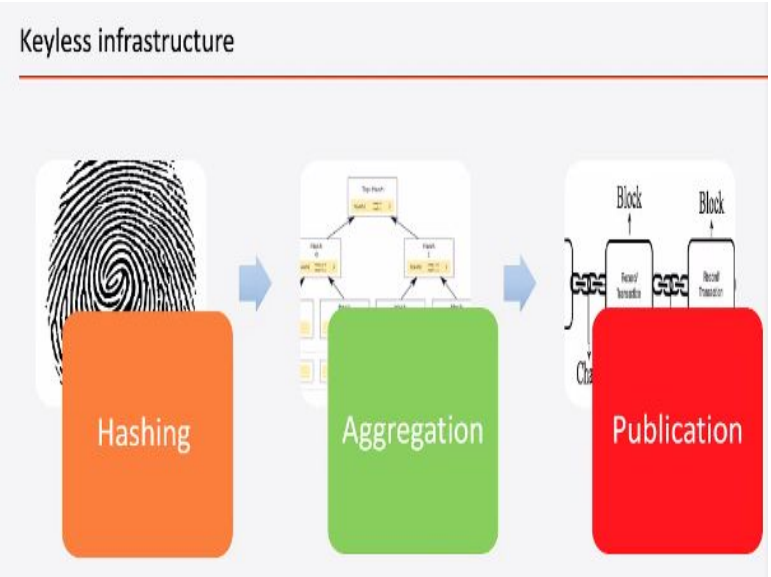
Cons

- Still uses Proof of Work
- Use ternary number system
- Crypto is not verified or tested: violates "Do not roll on your own crypto"
- Cannot handle smart contracts

It still uses proof-of-work. Uses a ternary number system, The cryptography being their own, there is a possibility that

the crypto is breakable. There is this idea in the cryptography parlance that do not roll your own crypto, you always use well-tested, well-validated crypto that has not been done and of course it does not have smart contracts.

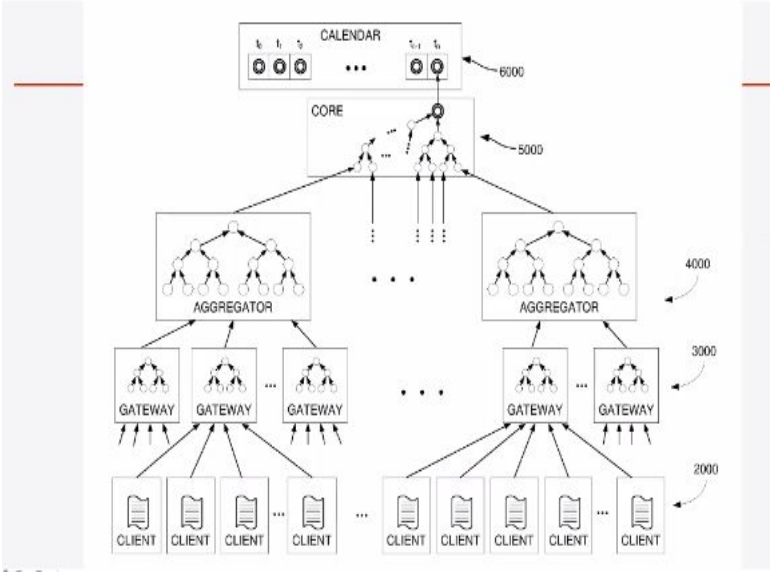
(Refer Slide Time: 53:52)



The other blockchain that we did not talk about in this course is the keyless signature infrastructure blockchain of Estonia. So it is actually an Estonian project that has gained a lot of traction and here the idea is that you basically depend completely on hash. There is no private key, there is no public key or there is no secret key and using the hash technology and the hash graphs which are kind of like Merkle trees, they actually have created a system where you can have integrity and time stamping of any document that is created in their government system.

So, we would very briefly look at what this is. So, it is based on the idea of hashing, aggregation of hash in the form of a hash tree or kind of like Merkle tree and then publication of root hashes. So what that means?

(Refer Slide Time: 54:08)



So, it means that say you have all these government organizations. Each organization will have some gateway and for every second of the day, any document that is created for e-governance, be it the creation of a license, creation of an identity, a registration of a property, etc., all that stuff are always being created right. So at any point in time, at every second whatever is created is hashed, all the documents are hashed and put into a hash tree at the gateway. Then for the next second again, the gateway will create another hash tree and percolate it to aggregators.

So it may be multiple layers of aggregation. At the end, all these different locations of the government and offices and everything, all the documents that are created the hash eventually gets you know rolled into a hash tree and then eventually those hash tree roots are again aggregated from various locations and eventually it goes into a core hash tree and then finally the root of that core hash tree is now put into something called a calendar blockchain. So for every second, there will be a single loot hash that gets into the calendar block chip.

So, now you have a kind of new block every second of the day that is permitted here and they are hash connected and then every Sunday or something, they will publish the latest hash. Now see what happens that suppose this is my property paper that was created at certain date, so I come back to the office to validate that this has not been tampered with. So what you have to do is they have to, by the way when they gave me this document and then the gateway hashed it, they gave me a token and this token basically contains the hash of the let us say this is my document.

In order to check whether the document hash is correct, what they have to do is they have to give me in the token they give me this hash, this hash and that is it. So when I come back, what they will do is they do not need these documents or these documents, all they need is look at my token and see whether it matches their version of my token and then they will take my documents hash and then this is in the token, so together they will create this hash. Then from my token, they will retrieve this hash.

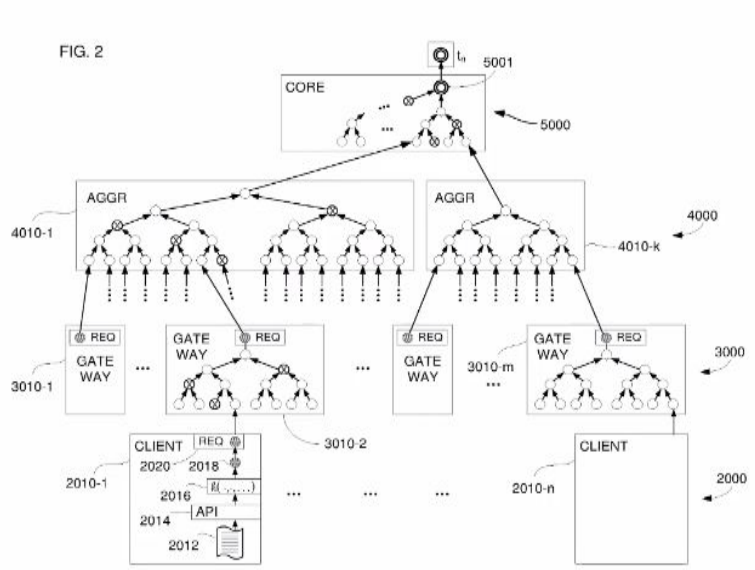
Then by this together they will create this hash and then they will check whether the hash here is actually matching this root hash. Now in order to check whether this root hash matches with the root has now being calculated when I bring back my document, they have to basically have the right set of you know complementary nodes hash values which will eventually lead me to that particular seconds calendar hash. So, integrity of this calendar hash has to be guarded critically.

Because if the integrity of this hash values are compromised, then certainly somebody could have changed some of these hash values as well and then accordingly change the integrity of the corresponding top hash value. So in order to ensure that none of the hash values that were stored for that particular second has ever been tampered with, integrity

of this hash value is very important and that is why this is actually a block chain with a hash connection between all the subsequent nodes.

In order for public trust, in order to enter enhance public trust, what they do is that every so often they will publish in all daily newspapers and all other channels the latest hash periodically. So if anybody has tampered with anything, then if challenged, people can say that on so and so date you told me that was the latest hash, now go and show me that that is still the case and then if they are matched, then that means nothing before that has been tampered with, so that is the idea.

(Refer Slide Time: 59:36)



Then idea is been implemented a very painstakingly throughout the country with every location having the gateways and then aggregators throughout the entire country's network. So, this is a very extensive project and it is actually taken care of by a company called Guardtime and it is a commercial company and they actually have implemented this very successfully. So everything in the e-governance in Estonia, so if you actually get time to look at the E-Estonia on Google, look at the use of this hash technology, this case either, or this is called a keyless signature infrastructure.

Because they are only doing hash, there is no public key, private key or anything and then a whole idea is based on the idea that the all government documents should be kept you know in a tamper-proof manner and there should be a way to prove that nothing in the government information system has been tampered with and therefore the trust of the citizen on the governance is much higher which is not the case in most countries because there has been lot of corruption.

The IT systems are centralized under the control of NIC or some other government organization and they could tamper with all this data and then when you go back with your version of the document, it might not match with what we are giving you and what they have in their system and this leads to a lot of litigation and all kinds of stuff and now certainly reduces trust of citizens in the government. In this case that is what they have achieved is that their trust, the citizen's trust in the government has been enhanced significantly by use of this technology.

(Refer Slide Time: 01:01:38)

KSI: Pros

- Privacy
- Publication of the root hash guarantees immutability
- No mining: Highly scalable
  - 10^{12} registration per second
- Quantum immune
- Indefinite key expiry

So the pros is that it is private because none of the data leaves the government, only the hash that gets percolated as well as the token gets only the complimentary hashes, so no data gets in the hands of somebody else. Publication of the root hash periodically guarantees the immutability of the calendar hash chain and that is what gives the trust. There is no mining involved and they are scalable. So they do actually 10 to the 12 which is basically a million millions which is a billion registrations per second.

The hash is quantum immune and there is no notion of key expiry because there is no key. So, to speak. So when you do a registration of a document or something you get a token and the token is basically the complimentary hashes in the hash tree.

(Refer Slide Time: 01:02:46)

KSI: Cons

- Private/permissioned
- Patented by Guardtime
- Smart contract not available
- Loss of token

It is private or permissioned in the sense that for finally the calendar blockchain and the entire hash chain system is actually provisioned and organized by the government with the help of this company called Guardtime. It is also patented by Guardtime, so that is a problem and smart contract is not available in this case because here there is no other thing that is being done for this system except for the integrity of the documents and there is possibility of people losing their token in which case they cannot go back and verify things.

(Refer Slide Time: 01:03:27)

Summary

So in summary, what we have seen throughout the class, of course we discussed something today that are not part of the class that is Algorand and KSI blockchain and also another thing that we did not discuss is the Zk-Snarks or the idea of Zcash which is built on top of bitcoin and the idea there is to use zero knowledge proof to actually give guarantees of proper you know anonymity rather than pseudoanonymity, but in general these are some of the concepts that we have discussed in the concept and we did not discuss some of them.

For example, we talked sometimes about the game theoretic sense why incentive mechanism works in keeping people honest and so on, but actually game theory people will analyze them as games cooperative or non-cooperative break games actually and then they will argue that this particular strategy achieves the Nash equilibrium and all this concepts and cryptographic sortition this idea we saw that the case of Algorand. We saw extensive use of Byzantine fault tolerance or BFT. Zk-Snarks is the technology of zero knowledge proofs behind the full anonymity of transactions in Zcash which we did not cover.

We talked about permissioned block chains and obviously un-permissioned block chains, etc. So there are lots of different concepts that are either covered or mentioned in this class and of course as I said from the beginning, this class is not about learning how to program a blockchain which you can learn very quickly with the help of online tutorials, here the idea has been always to actually give you a very good idea about how to conceptualize understand the differences between different blockchain technologies.

The salient differences so that you have a better knowledge of how to choose what is the most suitable blockchain technology for your particular problem or whether blockchain at all is the solution to your problem, etc. So in that sense, you know we have covered a lot of grounds, but of course there is a lot more to learn and you can probably have an entire master's degree program in blockchain which obviously in an 8 week course we are not covering, but you get hopefully some basic idea that we covered.

(Refer Slide Time: 01:06:39)

At the end of these slides, I have given a number of credit to various images we have used in this set of slides.

(Refer Slide Time: 01:06:51)

https://www.pinterest.co.uk/pin/5066618308753012/

https://chrispacia.wordpress.com/2013/09/02/bitcoin-mining-explained-like-youre-five-part-2-mechanics/

https://medium.com/@lhartikk/a-blockchain-in-200-lines-of-code-963cc1cc0e54

https://medium.com/hashtaagco/blockchain-say-what-8e16ed29e543

https://patents.google.com/patent/US20170033932

https://medium.com/iota-et-tangle/presentation-iota-tangle-2d6e63e3a70

https://www.bitcoinbeginner.com/blog/what-is-iota/

But also in this if you go there, you will find other interesting documents and blogs etc., that can help you to enhance your knowledge.

(Refer Slide Time: 01:07:00)

https://medium.com/@philippsandner/comparison-of-ethereum-hyperledger-fabric-and-corda-21c1bb9442f6

https://www.cryptocompare.com/mining/guides/how-to-mine-zcash/

https://www.criptonoticias.com/aplicaciones/zcash-celebra-primer-aniversario-impulsando-privacidad-intercambios-atomicos/

https://slack.com/apps/A0P92U35Z-crypto-currency-coin-market-cap

https://www.wired.com/story/how-to-set-up-twitter-lists/

https://www.vectorstock.com/royalty-free-vector/crypto-currency-zcash-silver-symbol-vector-19123605

https://atozforex.com/news/zcash-price-long-term-forecast-2025/

https://www.dreamstime.com/stock-illustration-thank-you-text-illustration-social-icons-tablet-computer-mobile-cellphones-cyan-digital-world-map-background-image48170847

Hyperledger Documentation

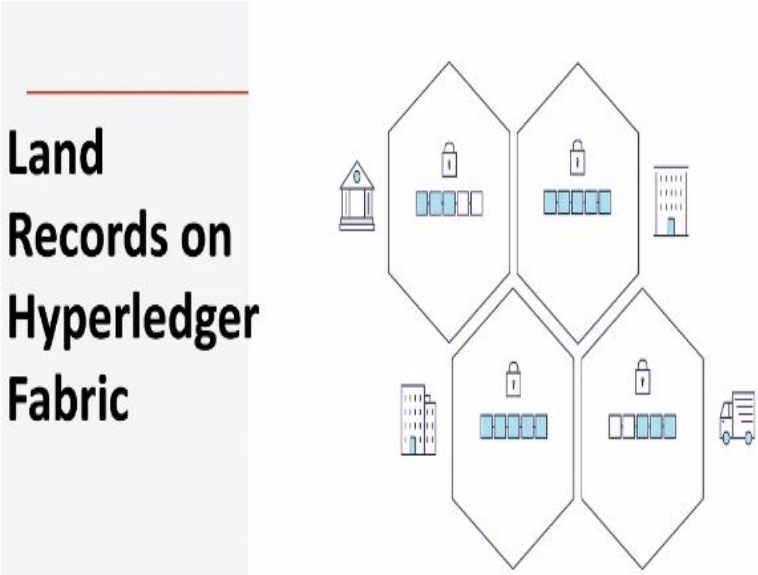
So when we come back, we will talk about some applications that we have worked on on the use of blockchain in e-governance and then after that we will talk about some final thoughts about blockchain and try to correct some of the misconceptions people have about blockchain. So, we will come back soon and till then stay tuned. Thank you.

Blockchain Technology and Applications
Prof. Sandeep K. Shukla
Indian Institute of Technology – Kanpur

Lecture – 28
C3I Center

Hello and welcome to the second part of week 8 of blockchain technology and applications on NPTEL. So in this part, what we are going to do is to expose you to a couple of applications that we have been working on using blockchain and as you will see that these are all e-governance applications and so the applications that we are particularly looking at is land record registry and the other one is on the health care information.

(Refer Slide Time: 00:43)



So, land records on Hyperledger fabric is one thing that we have done and I will give you a very bird's-eye view of what this entails. So, as you know that in India the states actually maintain the land records and usually there is a department of land records or the board of land records and this is also called board of revenues in certain states and then there is a registry and stamps department which basically registers any change in the land record.

Any kind of transaction on land records like somebody selling a piece of land, somebody is buying a piece of land or transferring the ownership of a piece of land. All these things have to be registered and this registration actually requires to pay registration fee. So usually these are 2 different departments and what happens in most places today is that these are 2 different IT systems that are maintained. The board of revenues or land records department maintains a different IT system and then the registry and stamps maintain a different IT system.

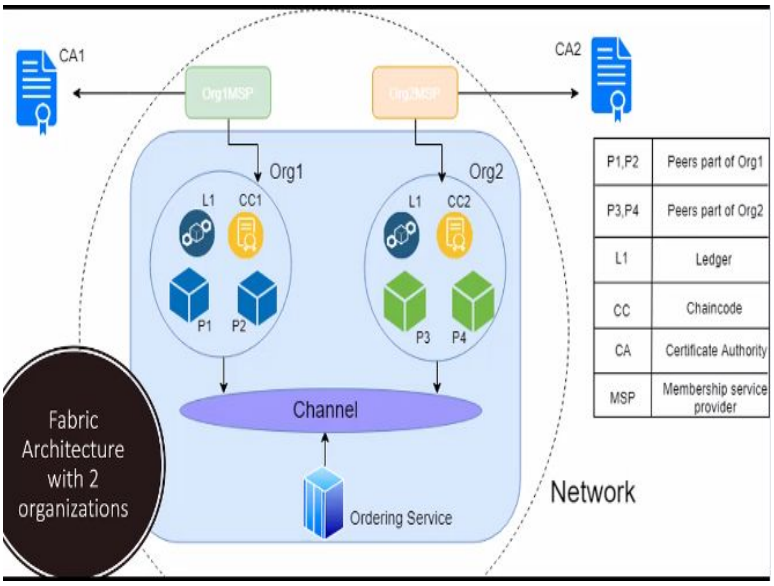
Then when people go to the registry pay their fees and do the registration, then certain information gets passed along to the land records department, where the final decision about the sale or about the transfer of ownership or other kind of transactions are adjudged to be legal or not and then the land records are accordingly fixed. So now what blockchain gives us here is if we use a blockchain and put all these different organizations on that blockchain.

Then there would be a seamless IT system, would remove any kind of lost in the middle kind of situations as well as any kind of manual errors that may occur when 2 different departments used 2 different IT systems, but more than that what happens today is that as I was saying in the previous part of this lecture is that trust in the government is very important for a functioning democracy and many times, in the land records, we find that there has been some problems in the maintenance of the land record, there has been unauthorized changes in the land record.

Also the document that is kept with the government and the document copy that the customers have may not match exactly. There may be some integrity violations while it is in the custody of the IT system of the government may be because of corruption or maybe because of mistakes. These problems can be solved if we use a tamper proof system in which any kind of tampering of the records would be easily detected.

So, with these in mind, we will show you how we go about doing this land records and as you can imagine, this is a problem that requires a decentralized ledger technology rather than a blockchain technology which supports cryptocurrency kind of applications. So Hyperledger is one possible choice for this kind of decentralized ledger technology.

(Refer Slide Time: 04:49)



So this is the idea and the idea that Hyperledger is pretty suitable for this kind of application comes from the fact that Hyperledger fabric architecture is such that you can define multiple organizations and you can have channels and these channels would allow you to keep information flow confidential and on the other hand the ledger that is being maintained by multiple peers belonging to different organizations are consistent and therefore this consistency and integrity of the ledger gives you more trust on the IT system that the government maintains and the citizens can be better served this way.

So, as you can see is that there may be multiple organizations for example this could be the board of revenues or the land record department, this could be registry and stamps department and they would be the organizations that are involved in this blockchain and there could be multiple channels or single channel as we will see and there may be different peers belonging to the different organizations even the ordering service, the members of the ordering service, the nodes of the ordering service could also belong to the different organizations.

The organizations might have problem trusting each other in terms of whether the one maintaining the records consistently or not, etc. So those kinds of mistrusts between the organizations can be handled in this kind of an architecture. So, we will do this with multiple channel architecture.

(Refer Slide Time: 06:58)

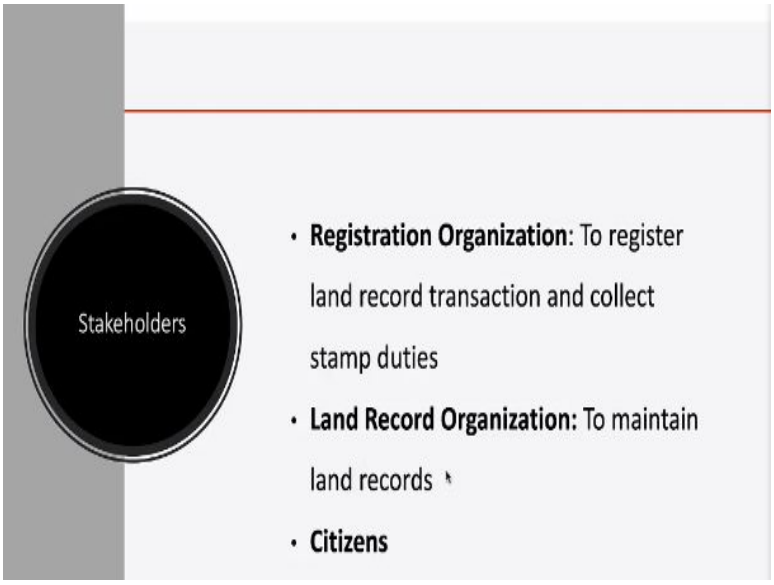
So, the problem statement here is that we want to design a decentralized system to capture land records. The integrity of the data should be maintained and privacy depends on what privacy requirements you impose. Certainly, right now the land record data is not really confidential, anybody can check for a piece of land who wants that kind of information, but the PPI or personally identifiable information, personal information should be protected. So, we

should not reveal the let us say Aadhaar number or PAN card number or bank account number of the person even though that may be in the system.

The verifiability is required because the owner should be able to verify his claim of ownership. So right now, it is a piece of paper or some kind of a digital document that they maintain with them and then the idea is that once he presents that to the board of revenue, board of revenue has to actually verify it to know whether this piece of paper is trustworthy or not and in many cases what may happen is that if the data has changed in the IT system of board of revenues, then the owner would be in trouble.

So, on the other hand if the owner has been forging the document in that case and has bribed somebody internally to change the record accordingly, then the rightful owner would be deprived from his rightful claim on the piece of land. So, the ecosystem here is the customers which are usually buyers or sellers and then the registration organization and the land record organization.

(Refer Slide Time: 09:11)

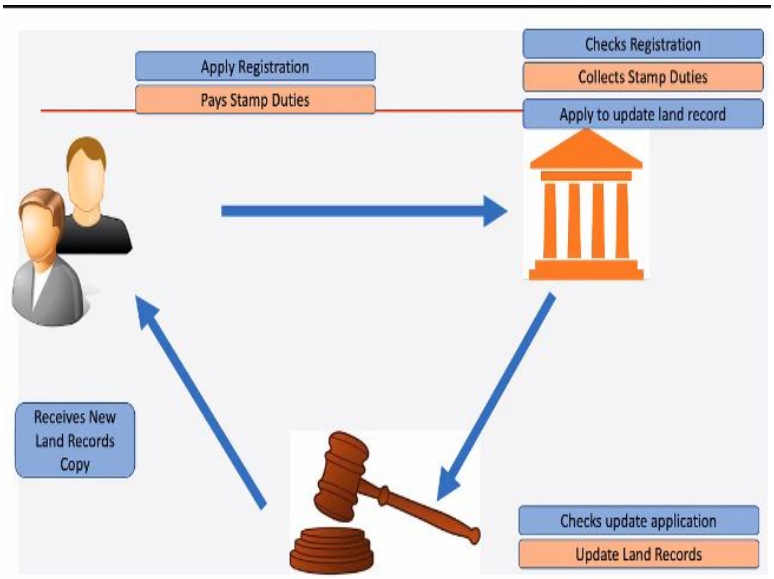


The registration organization is usually there to register land record transaction and collect the stamp duties and land record organization is there to maintain the land records. Citizens who actually own lands or want to buy lands or want to sell lands, they are usually the other parties that will be using this system. So, these are our stakeholders in the system.

(Refer Slide Time: 09:41)

So, land record management happens through this registry and update. So, registry basically is a transaction where a land transaction is registered, buying a land, selling a land, changing ownership of land, etc. etc. succession, these are some of the transactions that are registered at the registration office and then based on the registration, the land record has to be suitably updated. So normally, we know this process mostly as mutation where the data in the ledger of the government has to be updated accordingly to that transaction that has taken place.

(Refer Slide Time: 10:33)



So, the customer, in this case let us say a buyer or seller, in this case buyer obviously will apply for registration and pay for stamp duties and this is the registration department. So, the registration department will check the registration for completeness and correctness, etc., will collect the stamp duties and most probably do a KYC on the client and then apply to update the land record. So, at this point the document goes to the department of revenues or the department of land records and the land revenue court will make a decision.

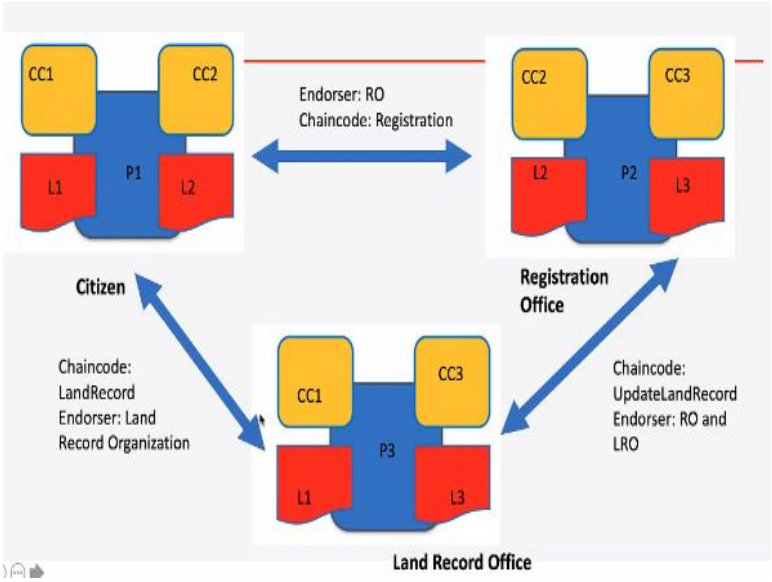
Usually it will give sort of a wide advertisement about whether anybody has any concern about this particular transaction and because sometimes the land is owned by multiple parties or when there is a succession transaction, maybe there are other successors of the original owner after his death the succession has been applied and the other should be able to get some time to know that their land is being served by somebody else, so therefore there is a time given.

After that the revenue court makes a decision as to whether to allow this land to be recorded to be in the name of the person who applied for this registration or names of the persons who applied for this registration. So, that is the checks update application process. It is a legal process and then the land record gets updated. Now this board of revenue has multiple hierarchies inside it with the local registers and so on and so forth, but they all can be part of the same organization as the board of revenue or land record department, and then the customer eventually gets a copy of the land record.

Now when we say if he gets just a printout that is not enough, right, because as I said that you know the printout can be forged and other things. So therefore, it has to be something more secure and probably digital and probably has enough information like as we said in case of the KSI blockchain in Estonia that there has to be a token which has some complimentary hash values embedded in it, so therefore it can be checked, etc. So, some such token has to be given.

Now, remember that this is not tokenization of the land, this is just a token to verify that the land record the copy that this person has is indeed the copy that was registered and updated, so that is the process.

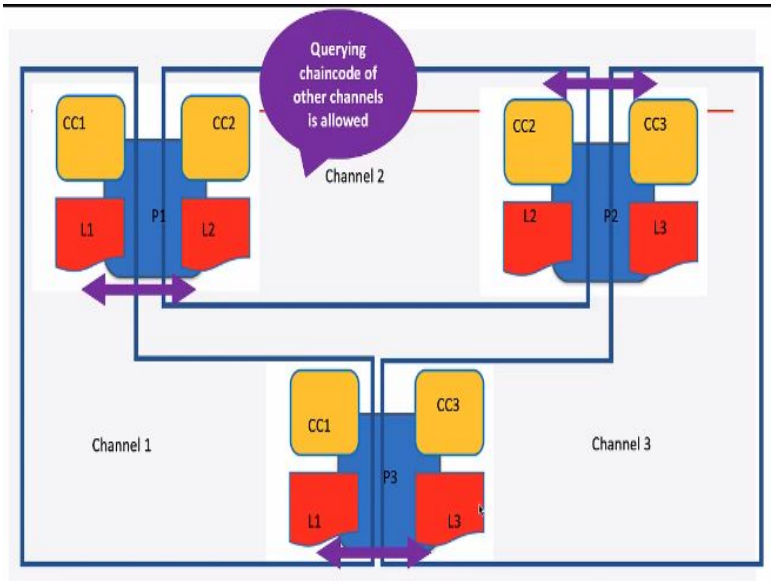
(Refer Slide Time: 13:57)



So, here what we can do is we can have all these different peers and they are working in different channels. So, the citizen will be basically a client which connects to peer to actually apply for registration and his registration has to be endorsed by the registration office and then when the registration office has endorsed, then it will update, it will send it to the land record office which will be making the necessary checks and make the decisions about the land records, you know validity of the registration and then accordingly it will do that and then send suitable digital proof to the citizen that the report has been properly updated.

Now, here you see that there are 3 chaincode involved. So, one chaincode is for the citizen to actually do its function, second is the land record that basically is part of another channel and a different ledger and this ledger is between the citizen and the registration department and then there is another chaincode that is between the registration department and the land record office, and then finally there is the L1 which is the ledger between the citizen and the land records office. So, this is the multi-channel solution.

(Refer Slide Time: 16:02)

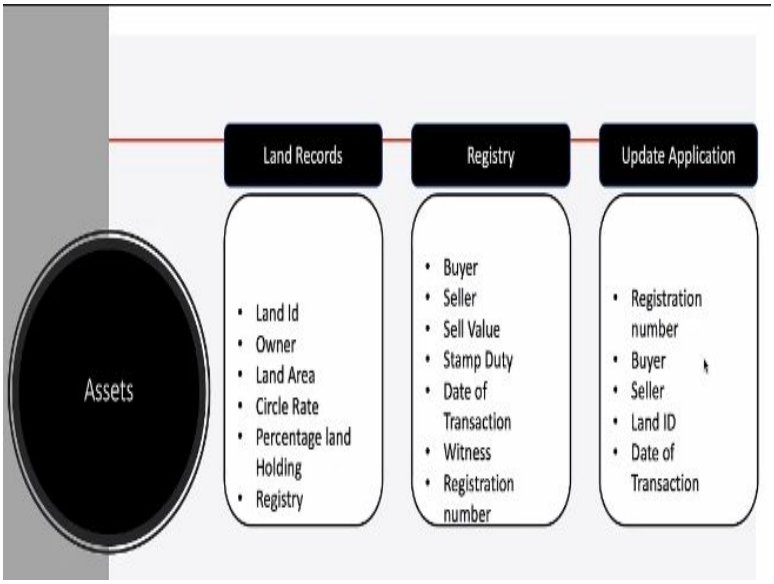


So here, you see that there are 3 different channels, one channel on which a ledger L2 that is between the citizen and it is basically keeping track of all the transactions and payments that the citizen does to the registry department, all the necessary documents, etc. Then the registry department when it endorses this transaction and then it basically uses the information that it glints from this transaction which is necessary for the land record department to do the investigation and accordingly update the record that is being shared through ledger L3.

Then the citizen and the land record department they actually are on this shared ledger L1, through which the citizen can always check the integrity and validity of the records that he actually owns. So, querying the chaincode of other channels is allowed, that is why this kind of situation is possible. So here, there is a query between this channel and

that channel, so the channel 1 and channel 2, similarly between channel 2 and channel 3 also there is a similar querying possible, and then between channel 1 and channel 3, so this is something that it is.

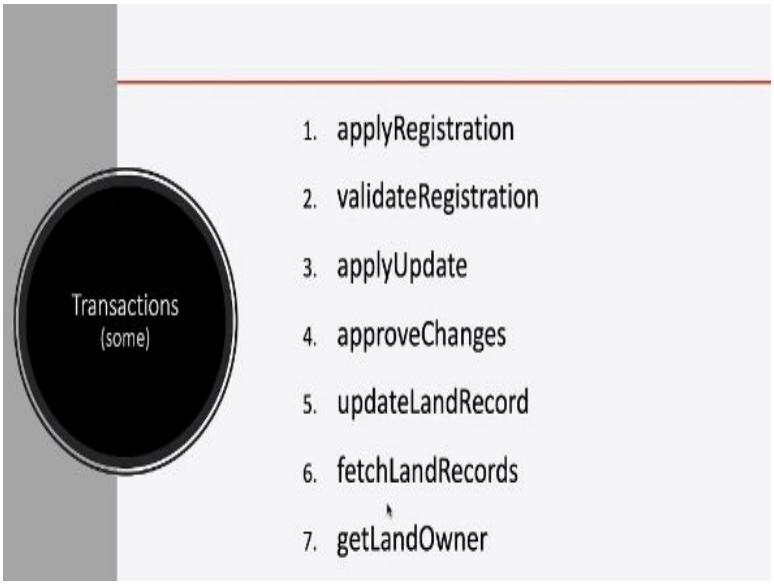
(Refer Slide Time: 17:43)



So, there are multiple assets and assets is a concept of the land records of the Hyperledger. So we have identified three asset types. So, one is land record, another is the registry, and another is the update application. So land records obviously we will have a land ID, normally like in Uttar Pradesh, the land ID is actually Aadhaar like unique number for every piece of land owner. That would mean that owner's name, there may be Aadhaar number, maybe PAN number, etc. etc.

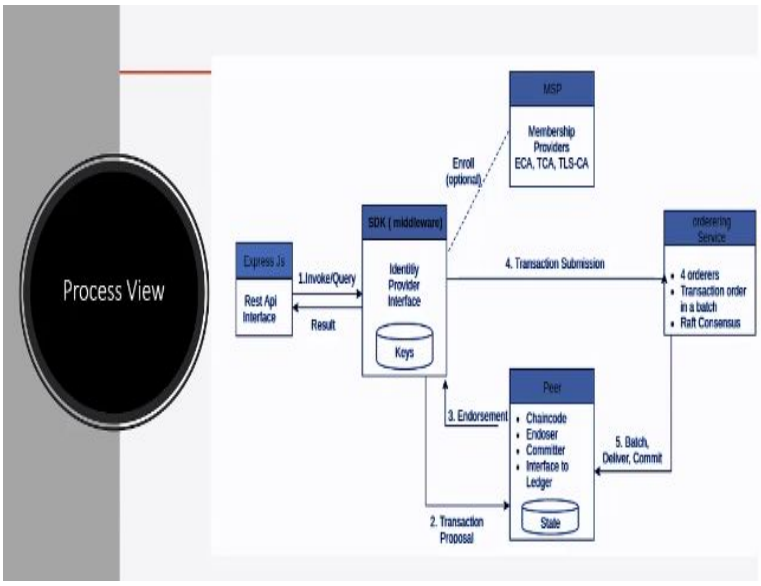
The information about the land, the land area circle rate, percentage land holding, this kind of stuff and registry related information. Registry on the other hand will have the buyer, seller, sell value, stamp duty, date of transaction, witness, registration number and all that information. Then update application will have the registration number which is basically being sent from the department of registry to the update land record department. So this kind of information will be there.

(Refer Slide Time: 19:06)



So, the transactions that are designed for this are applyRegistration, validateRegistration, applyUpdate, approveChanges, updateLandRecord, fetchLandRecords, getLandOwner, etc. So, applyRegistration is done by the citizen. ValidateRegistration is done by a transaction initiated by the registry office or RO. ApplyUpdate is also done by RO and then approveChanges and updateLandRecord these are done by the department of revenue or land records department and then fetchLandRecords and getLandOwner, this kind of transactions are done by citizens.

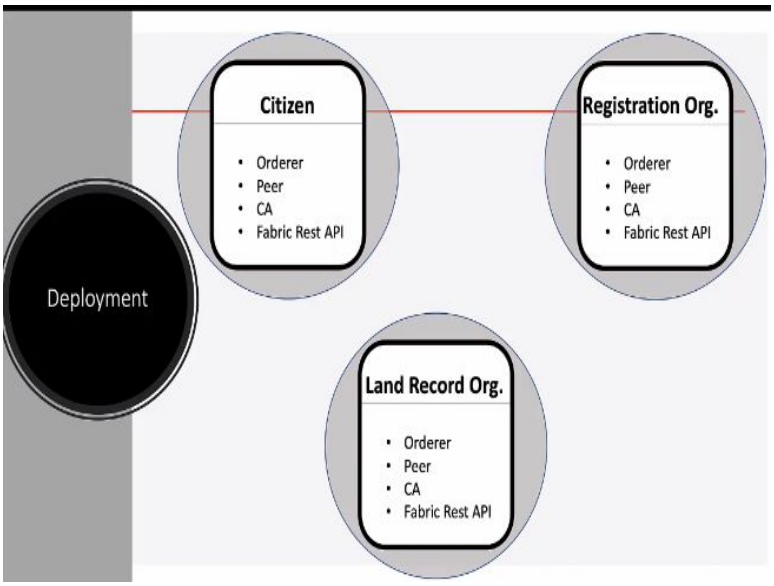
(Refer Slide Time: 19:58)



This is the process view. So, the process view basically says you know how this software actually was implemented. So this shows the peer and the SDK, this is the identity service provider which basically maintains the digital certificates or keys for all the individuals and all the servers that are you know participating in making transactions because each transaction has to be signed and then each endorsement has to be signed and that digital signature requires the digital identity.

Then this basically may depend on some membership provider like it may depend on a certification authority or some other kind of authority to derive the identity and then there will have ordering service which orders you know Byzantine fault tolerant way or crash tolerant way for making sure that there is an agreement on the order of transactions snd this is the basic structure of this implementation.

(Refer Slide Time: 21:24)



So deployment will be that registration organization will have one peer, which is an orderer, there will be a peer in the registration organization. There would be a CA for the registration organization and then it should have the rest API for interacting or invoking transactions. Same thing with the land record organization and the citizen. So that was sort of like a very high-level view of what a land record or land record registration and transaction system implemented in Hyperledger.

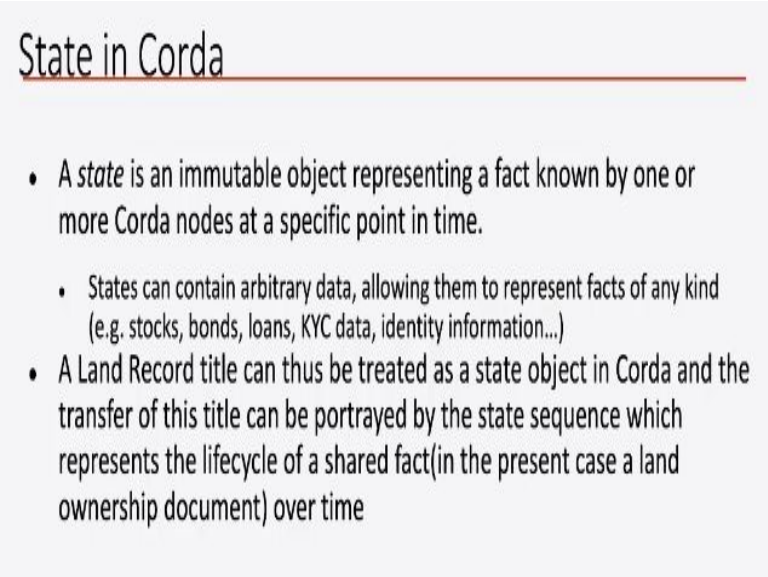
Now I want to show you another alternative way of doing this is to Corda because we said that Corda is another example of a decentralized ledger. So it is possible that one would prefer Corda over Hyperledger, but one has to actually first look at not only the convenience of designing the system because Corda as I discussed before is more geared towards the contractual obligations and contract obligated transactions, so that the contract requirements are fulfilled in the financial domain, but that does not mean that the Corda cannot be used in another domain.

Same thing as said before is that Hyperledger can be properly designed to actually do what Corda does, whereas Corda is more focused towards a particular sector, Hyperledger is more generic, but the choice has to be made based on the convenience of the architecting the system given the primitive conceptualizations of various functionalities that are available in that particular platform. Also once you implement it, you have to do proper performance testing, and to do performance testing you have to basically create a lot of transactions.

Random database transactions which are similar to the real world usage of this similar system and also the transaction rates and everything has to be tuned so that it kind of mimics the transaction rates and the inter-transaction arrival time, etc., for the real-world application in which this is targeted and then you have to see what the transaction throughputs are, what is the latencies are, then also you have to look at considerations such as security, ease of use, etc..

Then you have to make a choice between whether you want to implement this on Hyperledger or in Corda or in some other like maybe a private Ethereum. So these kinds of decisions you have to make or quorum which is a private Ethereum variant you have to do based on those things.

(Refer Slide Time: 25:16)



So here is our attempt at using Corda to do this same thing as we said is the land record transfer using Corda. Now in Corda as we said is that one of the primitive concepts is the state. State is an immutable object which represents a fact known by one or more Corda nodes at a specific point in time. Remember the transactions in Corda take states as inputs and produces states as outputs. So, states can contain arbitrary data, so that they can represent facts of any type like stocks, bonds, loans, their ownerships, KYC data, identity information and all that stuff.

So land record title can then be treated as a state object in Corda and the transfer of this title can be portrayed by a state sequence in which before a transaction the state who recorded who owns the land at this point in time, what are the other owners, and all that stuff and then as the transfer takes place if it succeeds, then the next state will be recording the name of the new owner and other relevant information.

So in that way, any land will be associated with a state so at any point in time at present or in the future would be actually a state information about the land, right. So, the virtual representation of a piece of land is actually a state object in Corda.

(Refer Slide Time: 27:10)

So, the participant, Corda nodes in this case would be buyer, seller and the government and the government can create a new land record document with specified owner. Buyer can request to buy a specific land record basically in the requested state change associated with the virtual representation of the land. Seller can sell or transfer ownership of the land record and thereby basically changing the state that is representing the land.

(Refer Slide Time: 27:50)

So land record would have as I said, even in the last case also we saw that land record would be having land ID, the owner name ID and depending on whether a piece of land can be owned by multiple people, it could be actually a list

of owners, previous owners which should be also a list of previous owners and there may be other relevant information about the date of transfer in the past from one owner to the next owner and so on so forth.

Buying requests would be that is the list of potential buyers. So whenever somebody makes a buying request transaction, it may go to an intermediate state where a list of potential buyers could be recorded.

Refer Slide Time: 28:42)

So in Corda, we also talked about the notion of a flow, right. So flow is basically automates the process of agreeing ledger updates, right. So flow is basically orchestrated the different activities at different places in the Corda system which would have to take place in order for the update of the states and the ledger. So, there would be 3 flows here that is create land record flow, request ownership transfer flow, and then transfer ownership flow. This is kind of you know similar to what we were doing in Hyperledger where we had 3 channels, so here we have 3 flows.

So, create land record flow would be a buyer and then government would be participants and it can be initiated by government only. Request ownership transfer flow can be initiated by buyer and the participants are seller, buyer and the government. Transfer ownership flow would be also seller, buyer, government and can be initiated by seller only. So, as you can see that this is when the government actually starts a new land record, this is when a registration request is made and this is when the actual mutation happens, that is when the actual land record is updated.

(Refer Slide Time: 30: 20)

System Design - Contracts

- A **contract** takes a transaction as input, and states whether the transaction is considered valid based on the contract's rules. In our application the contract should check for the following
- Create Land Record Contract
  - Zero input state & One output state
  - Initiated by Government
  - Two signers namely, buyer and government
- Request Ownership Transfer Contract
  - One input state & One output state
  - Initiated by any buyer
  - Three signers namely, buyer, seller and government
- Transfer Ownership Contract
  - One input state & One output state
  - Initiated by current owner
  - Three signers namely, buyer, seller and government

So, to contract in this case is basically will take transaction as input and states where the transaction is considered valid based on the contract's rules. In our application, the contract should check the following. The create land record contract will check that there is zero input state and one output state. It should be only be enabled to be initiated by the government and there should be 2 signers, namely buyer and government. Request ownership transfer contract would be one input state, one output state, initiated by any buyers and 3 signers namely buyer, seller and the government.

Transfer ownership contract would be one input and one output state, initiated by the current owner and there should be 3 signers namely buyer, seller and the government. Now this is highly simplified, I mean we are not exactly going by a specific state government's exact procedure because we are not fully privy to the entire process that the government follows, but it can be customized according to the government rules and processes as in case we actually do it for a specific government. Here, we are just giving you some basic idea about what it might be like.

(Refer Slide Time: 32:00)

System Design - Consensus & Notary

- Determining whether a proposed transaction is a valid ledger update involves reaching two types of consensus:
- Validity consensus - this is checked by each required signer before they sign the transaction
  - In present application the signers would be buyer, seller and the government nodes
- Uniqueness consensus - this is only checked by a notary service
  - Notary clusters can have several nodes that can be owned by government and NGOs(acting on behalf of citizens) which can act as a neutral third parties

So, whether a proposed transaction is a valid ledger update would require two types of consensus, right, one is of validity consensus and other is the uniqueness consensus. So, validity consensus is checked by each signer because before they sign if the buyer says that you know, I am getting this piece of land from this seller, then buyer should first verify that this is a valid transaction, government should verify this is a valid transaction and buyer must verify that he or she actually owns the land and then actually sign this validity consensus.

Here, the idea is that hopefully the government actually will be the most important one here because you know if the buyer is not telling the truth, the government has had to actually first do the proper investigation. So this cannot happen like overnight, the government has to put out an ad about this update and then anybody who has any kind of you know reservation against this update should be able to go to court of revenues and put objection and so on. So once all that is over, only then the government will sign.

The uniqueness consensus has to be checked by a notary service. So we talked about notary service in the context of Corda. So notary basically checks whether this particular state of land currently is unique in the sense that somebody did not use the same to make another update transaction before, right, because then the land has been already updated in somebody else's name, then this seller is not really the correct owner and this should not be allowed right.

So, this notary service could be actually several nodes that can be owned by the government and maybe NGOs acting on behalf of the citizens which can act as neutral third parties. So therefore this is what actually the court of revenue does, and then once this thing has been actually cleared only then the uniqueness consensus will be achieved. So that was a Corda version of the land record, very superficial view of how this can be done.

(Refer Slide Time: 34:49)

Then finally, I want to talk about the electronic health record application and again we are using Hyperledger fabric here.

(Refer Slide Time: 35:08)

So the problem statement here is that, so let me first explain what the problem is. So when we go to a hospital, normally what happens is that the hospital checks us in. Then it takes some vital information like blood pressure, temperature, weight, etc. It records them and then it sends us to the actual doctor. Doctor makes whatever checkups that he has to do and then he might order various tests right, pathological tests, blood tests, urine test, etc., and then it has to go to the pathological laboratory.

When the pathological laboratory collects samples, they return all the results of the tests, then you go back to the doctor and then the doctor takes a look at this information and then accordingly may prescribe. Now doctor may also prescribe before the tests are done for temporary relief, but may actually make a more specific prescription after that

information comes. Anyway, so once you get a prescription, you have to first pay the hospital for having seen the doctor, etc., but then you go to the pharmacy.

The pharmacy then fills your prescription and then gives it to you, and then if you have insurance, then the insurance has to actually get involved during your payment to the hospital and maybe the payment of the medicine, may be payment at the pathological tests, and also they might have to be initially involved when you actually first make the appointment that they are agreeable to this particular location, this particular hospital or they actually make cashless payments or reimbursement based payments whatever, all this information has to be gathered in the beginning also.

So, this is a pretty involved multi-party activity that happens. Now usually what happens is that we have all these documents and either in a more advanced hospital, this documents, this information that is taken about a patient like his vitals on a particular day, his pathological test reports, his doctors collected information, the information about the prescription being made, the state of the insurance payment all that stuff, these are either maintained on a central database by the hospital system or it is maintained in bits and pieces at various systems like in the pharmacy, in the pathology lab, etc. and they are not integrated.

There in the first case, it is integrated, but it is centralized and then when you go to a new hospital system getting that information quickly from one hospital system to a different hospital system would be time consuming, either through some email or something they will send your past history to the new hospital or you have to go and collect printed copy and then bring it to the new hospital. So this is quite cumbersome.

But bigger issue is that all the patients information if it is maintained centrally in a hospital in an IT system, then anybody who has access to that IT system, be it a doctor, be it a nurse, be it a front desk person, etc., the pharmacist or pathologists they all get to see all your information, that gives a privacy concern and in the United States there is a law called HIPAA which is very strict law about patient health information privacy protection act, HIPAA, and that kind of act is needed in India as well.

So if that kind of an act comes, then hospitals cannot be just leaving this data you know unprotected, cyber unsecured on some server or on a cloud server and then be happy about it and the patient certainly would be happy about it because all that information what disease they have and what kind of you know maybe certain diseases they do not want everybody to know, their neighbors to know, etc., all that information should be protected by law. So another issue is that, one is either the data is in a centralized database in the ownership of the hospital which is not already a very good thing.

Then on top of that there is privacy concern. Now data that is regarding the patient should be data principle is the patient himself and then if hospital becomes a fiduciary which maintains that data, then hospital will have a big onus once a data privacy act comes into existence that the hospital has to be always afraid that if the data gets leaked, then the patients are going to sue them and they might have other legal liabilities. So therefore, we need a system which is decentralized and to capture medical records.

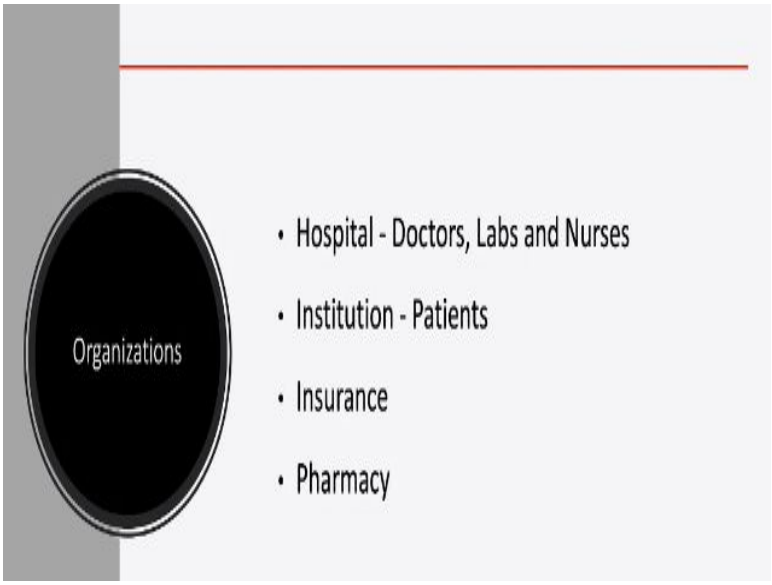
Also, the other issue that I forgot to tell is that in a centralized system like that anybody can go and change your data. So there have been movies about it that some patient did not have diabetes, but the data showed that he has diabetes. So when he was unconscious in a hospital in an emergency situation, the doctor gave insulin in his fluid that intake and then they had an insulin shock and they died. I do not know about a real case like that but this kind of scenarios also worry people that if the integrity of the information is not maintained, then anybody can manipulate that information and get you into some trouble.

So therefore, the question is that the centralized ownership is not good either for the data fiduciary that is the hospital because they have the huge liability for the patient because patients confidentiality could be breached and also if the data integrity is manipulated, then the patient may be actually in danger. So to solve all this, one has to consider using a decentralized system to capture the records, the integrity of the data should be guaranteed and then patient privacy should be preserved.

That is if the patient as the data principle allow somebody to look at the data, only then that data can be looked at, otherwise it should remain unreadable to everybody, that is it should be remaining encrypted. So now there has to be a solution in which the patient can selectively show the data based on a need-to-know basis. For example, a pathologist does not need to know patients vitals and other information or pharmacists does not need to know even what actually is wrong with the patient because their job is only to fill the prescription.

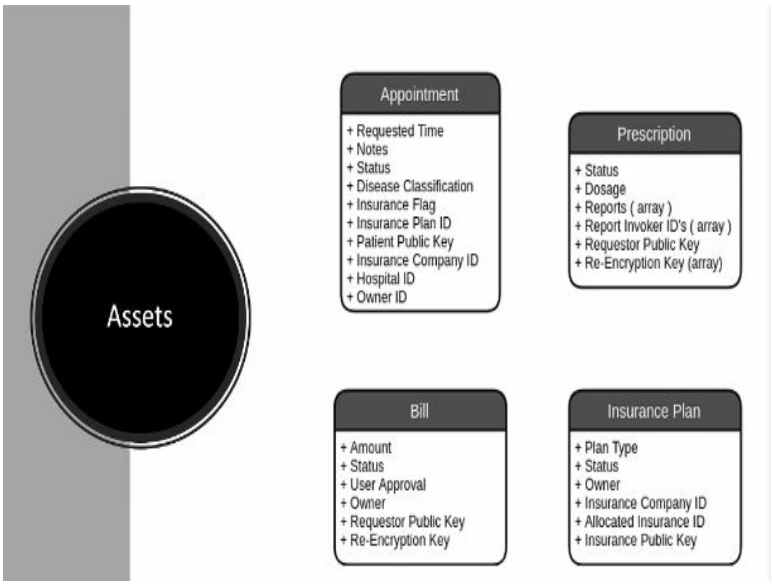
Then the integrity of the prescription is important because if they get tampered with, then the pharmacists will give the wrong prescription to the patient. So, we want a complete healthcare information system in which all these problems are addressed and then it is a decentralized system, data integrity, historical data about the patient that's integrity should be maintained, confidentiality of the patient should be maintained and patient should have the full control on who sees the data and for how long.

(Refer Slide Time: 44:18)



So, the design that we have is that we have organizations like hospital where there are multiple parties which interact with the system like doctors, the pathology labs, the x-ray labs, the nurses and the front desk people. Institution that is patient and then the institution means if it is an institutional hospital, insurance companies and the pharmacy.

(Refer Slide Time: 44:54)



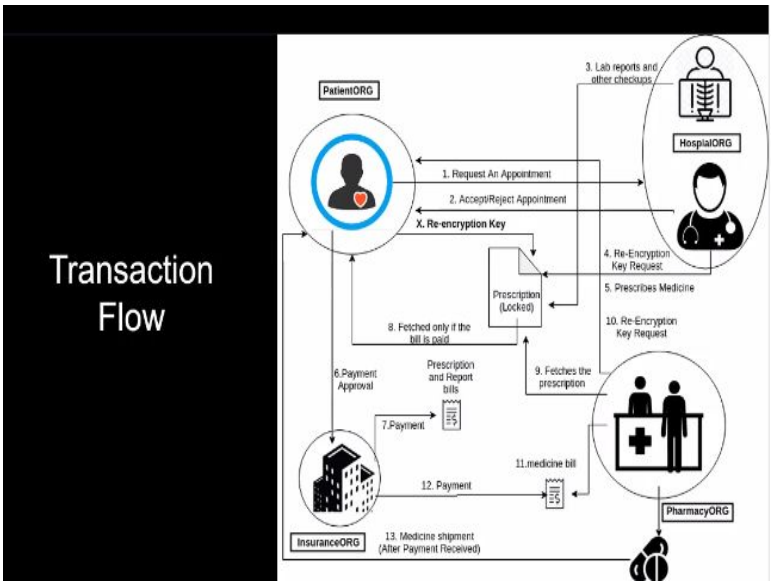
So, there are multiple different assets involved here. So appointment is one asset, prescription is another asset, bill is another asset and insurance plan is another asset and there may be some few others you know, I am showing as very simplified version. So when the patient books an appointment and instance of the appointment asset is created, which is associated with the patient and then when the patient comes and sees the doctor, then prescription is created and it is actually encrypted with the public key of the patient and therefore only the patient can see this.

But in this case you can also use something called a proxy re-encryption by which the patient can generate a proxy key that allows say insurance or a pharmacy to actually look at its prescription information which is the one of the vital personal information that needs to be confidential.

(Refer Slide Time: 46:04)

So, different transactions like requestAppointment, acceptAppointment. RequestAppointment is done by patient, acceptAppointment by the hospital front desk, getAppointment status can be done by the patient, generatePrescription can be done by the doctor, getPendingPaymentRequests is by the patient, registerInsurance is by the patient or the insurance company, acceptInsuranceRegistrationRequest is doubly endorsed by the insurance company and probably the hospital, allowPrescriptionOthers is by the patient, getBillStatus by the patient, getInsurancePaymentRequest is by the insurance company and so on and so forth.

(Refer Slide Time: 46:53)

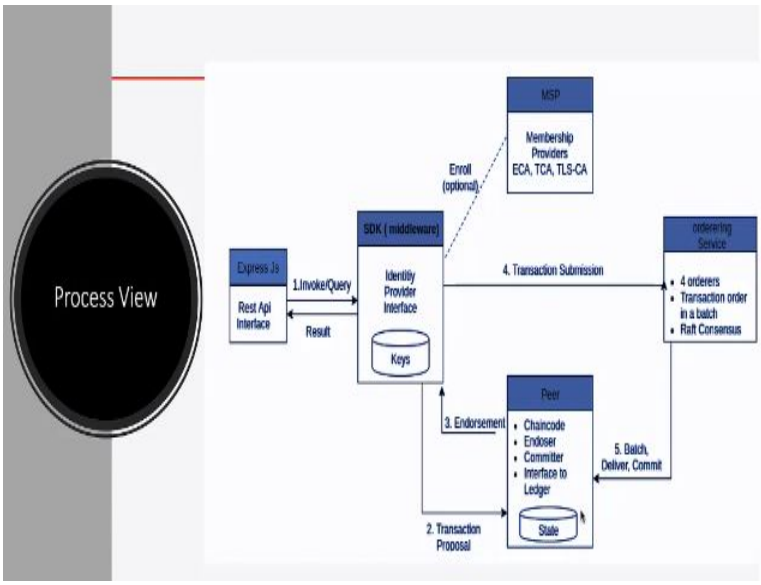


So, the transaction flow we are just showing a very simplified picture of the transaction flow. So you have the patient, you have the hospital, so there are multiple organizations right. So patient organization, this is a hospital organization, insurance organization and pharmacy organization. Then idea is that the patient makes request for an appointment and then either gets accepted or rejected the appointment. When it is accepted, the patient goes to the hospital and then his data is basically sent to the doctor.

Then the doctor sees him and then may actually request lab reports and other checkups, but once that lab reports and other checkups come in, then the prescription is created and this prescription is locked by the public key of the patient, and then when the pharmacy needs to see the prescription, the patient basically gives a re-encryption key, it sends a re-encryption key to the patient and the patient sends re-encryption key. Then this re-encryption key could be time bound so that the pharmacy can only see, only unencrypted for some time, and then it can fill the prescription.

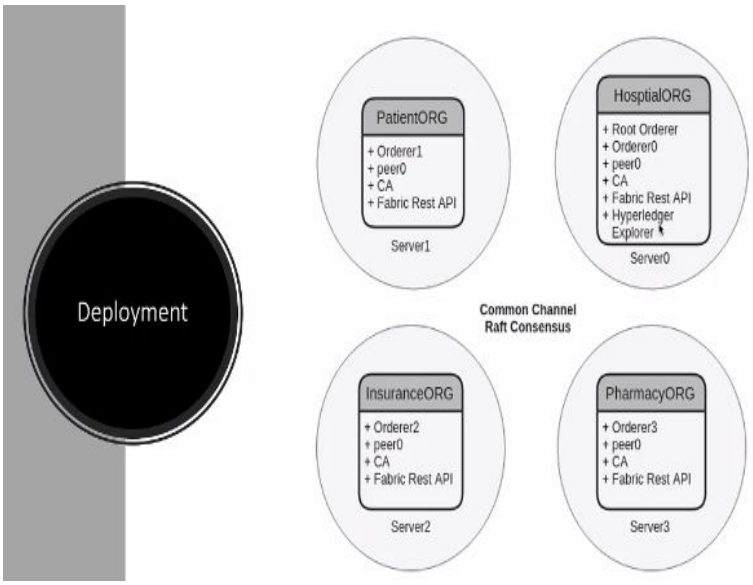
The hospital and the pharmacy, etc., will interact with the insurance as well as the patient, so this is pretty. Even in the simplified form, this is a pretty complex flow.

(Refer Slide Time: 48:35)



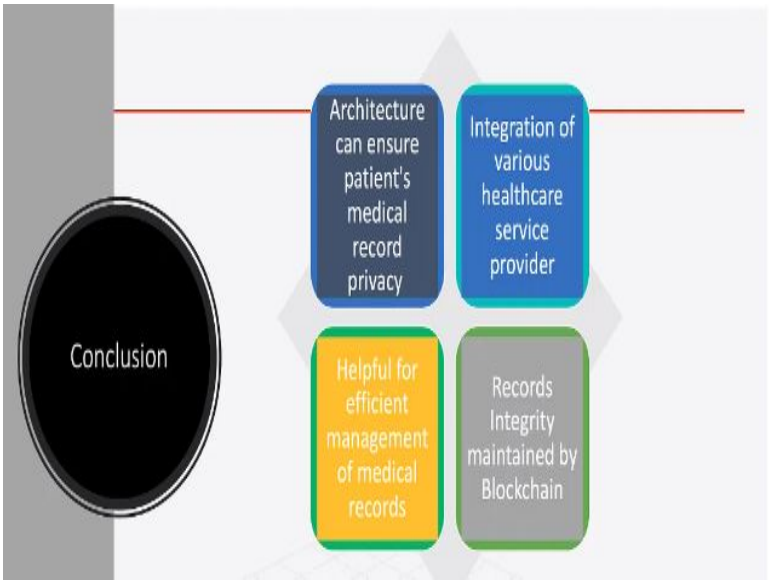
This is a process view which have peers, which has membership service provider, may be a some kind of a certification authority or if it is an institutional hospital like an IIT hospital, then it could be the institutional LDAP or some other way of giving membership proof or identity to the patients. Then you have ordering service which is maintained by the different organizations involved and then the peers.

(Refer Slide Time: 49:10)



So, the deployment view is that the orderer is maintained by all, so there are 4 orderers in this simple implementation which work for a crash tolerant Byzantine fault tolerant consensus and so it requires at least 4 orderers. So here, I have orderer belonging to each organization, but the root orderer or the lead orderer is under the control of the hospital. Then you have different peers associated with in each of these and then you have the CA and then the fabric rest API for in for transaction interaction. Then the hospital can also use a Hyperledger explorer to look into the ongoing activities through this entire system.

(Refer Slide Time: 50:14)



So, this architecture can ensure patients medical record privacy, integrate various healthcare service provider into one simple decentralized system and they can maintain a decentralized ledger. The prescription is always in encrypted form on the ledger, so only the patient has the ability to unlock it for some particular organization. So, it is helpful for efficient management of medical records and records integrity is maintained by the blockchain.

So this is a very simplified view of what one can do with this third-generation blockchain or distributed ledger technology for creating multi-organization integration of information systems with the property of information integrity and with the ability to transact across different organizations by maintaining a decentralized but shared ledger among all the different organizations. So, when we come back, we will talk about some of the final ideas and thoughts about this course as well as some misconceptions about blockchain that would actually help us conclude this 8-week long course.

Obviously as you know that after this, you have a long way to go, you have to actually learn at least one or two individual blockchain framework, how to use them, how to develop projects based on them which was not possible within the time frame we have and also the goal of my course was not to actually do this individual blockchain, you know nitty-gritty details and how to develop or program smart contracts etc.

So for that, there will be other courses, but this one is to get you a very conceptual standing or you know clarify your concept about what blockchain really is, what this technology is good for, where it is not applicable and where it is applicable, and how it is applicable. So, see you in the third part of this week and then we will discuss the final thoughts. Thank you.

Blockchain Technology and Applications
Prof. Sandeep K. Shukla
Indian Institute of Technology – Kanpur

Lecture – 29
C3I Center

Welcome to the part 3 of the final week of blockchain technology and applications on NPTEL. So in this last part, I want to talk to you about some of the misconceptions I have seen while discussing about blockchain with various people including the government agencies and I want to dispel some of these myths, so that you know you are also aware of those misconceptions and you know how to address them when you discuss the blockchain.

(Refer Slide Time: 00:35)

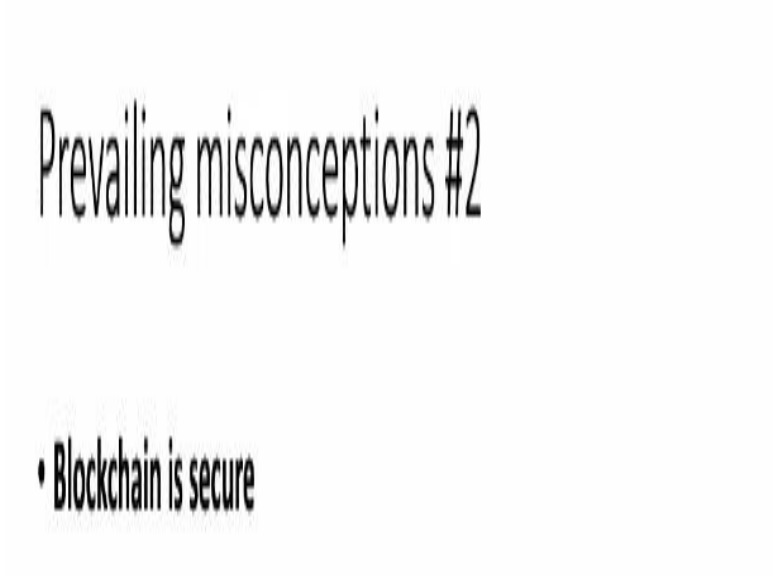
So, I have written down some of the misconceptions that I have been hearing about. So first one is blockchain is like the Internet, you need to build a network infrastructure around the country or in a state, so that everyone can use that infrastructure to do their e-governance and other applications, and I have seen this in Tamilnadu government's brochure where they are claiming to be building a statewide blockchain through which they want to provide e-governance services. Now, the problem is that blockchain is not like the Internet.

You do not have to prebuilt a blockchain on which all kinds of data will be thrown in. You have seen that you know we did for example a blockchain solution architecture for say land records, another one blockchain solution for healthcare, and they are quite different. Their assets that are involved are quite different. The transactions are quite different. Some have multiple channels. Some require single channel. The rest APIs for accessing, making transaction requests and so on are quite different.

So privacy requirements are quite different. So it makes no sense to build a blockchain, have nodes all over the state or region or the country and say that okay here is a blockchain, now you can build an application on top of this, it does not make sense. Each application needs to be considered in its entirety and what blockchain solution works for it will depend on the problem, its requirements, and then accordingly you have to create their solution.

There are multiple different platforms that are available when most of them are open source, so there is nothing hidden in that and you can always use them to build your application. So, this concept that you know we build a statewide blockchain or nationwide blockchain and then any kind of application will be built on top of it makes no sense.

(Refer Slide Time: 03:09)



Second misconception I keep hearing is that blockchain is secure, which means that they assume that tamper resistant data repository is same as secure, but we have seen in this course while discussing for example the Ethereum for example as soon as smart contracts came in, the number of vulnerabilities that were exposed were a very large right and we discussed some of the tools that people have developed to check the smart contracts for security vulnerabilities and so on.

So blockchain by itself by definition or by construction is not secure. What blockchain guarantees is that if you have a proper mechanism for transaction validation, if you have a proper mechanism for building consensus, then you can create this chain of blocks and then when the chain of blocks is sufficiently long, then going back and changing one information or more information in a particular block is going to be so computationally expensive that you can assume that it is going to be computationally almost impossible.

So, therefore you are going to see data in the grid. Now in case of block chains that are not based on let us say proof-of-work kind of consensus mechanism where the computational cost is not prohibitive, but they have probably something else, for example proof of authority or proof of stake where somebody unless a large number of high stake individuals in the blockchain participants do not collude and want to attack, the integrity cannot be easily tampered with.

In case of permissioned blockchain, it is secured by way of the fact that identities of each participant is actually established and therefore any kind of tampering attempt will be actually figured out because it will take long time to actually go back in time and redo every transaction except the transaction that you want to say eliminate and not get caught, because your redo identity is associated with that identity. So integrity is something that is kind of part of the reason why blockchains are used but does not make it any secure, for example against DDoS attack.

Against exploiting software vulnerability in your smart contracts to do things which would be otherwise considered from the blockchains perspective this is legal because this is acceptable because transactions are going through according to the smart contract and everything, but the smart contract itself at the bug and therefore somebody is using it to do the bug. For example in the DAO attack, they used a bug in the security in the smart contract that allowed somebody to change the owner of an account and that is why the attack could happen. So, declaring that since I am doing blockchain I am secure is also a very wrong idea.

(Refer Slide Time: 07: 34)

Now another misconception is that since I am using cryptography, no matter what cryptography I am using, it guarantees that the blockchain data integrity is kept. Now, this again depends on the cryptography and the strength of the cryptography you are using. Of course, if you are using a cryptographic hash function for example that is so far known as you know collision resistant like SHA256 or SHA-3, you can be more or less feeling comfortable about the data integrity.

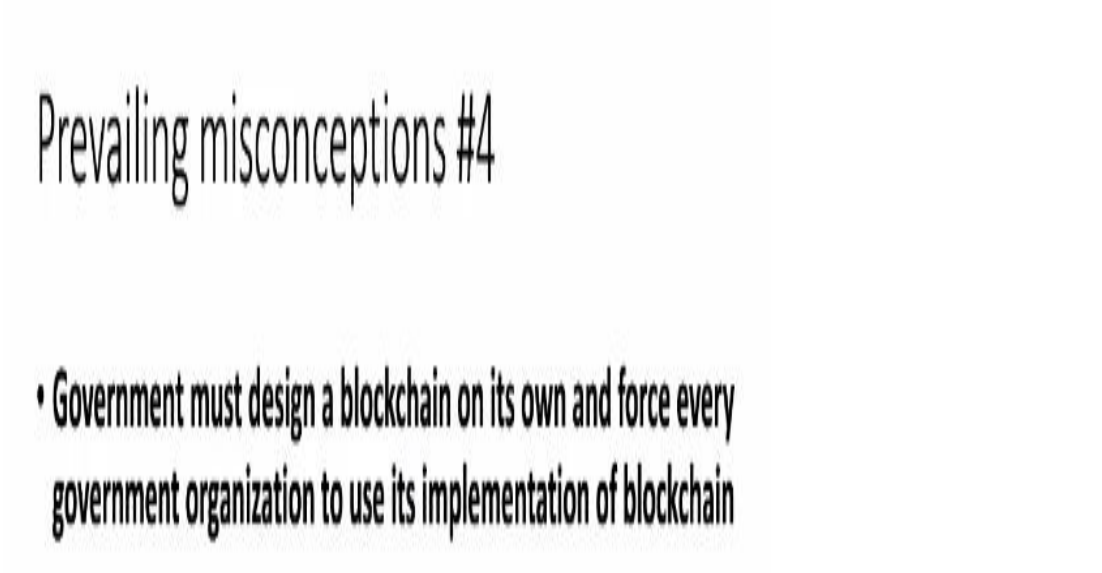
But we do not know in the future nobody would come up with a way of finding collision like SHA-1 was considered pretty secure MD5 was considered secure, was used for some time, but then people found ways to do collisions in them. Similarly cryptographic signatures because you are using RSA for example or and some other public key cryptography, you can assume that if you are using the proper key size and everything, it is not possible to forge signatures, but we know that RSA 1024 bit long keys are no longer considered very secure.

Also there could be other things like side channel analysis on your machine on which you are doing the cryptography and then the public/private key maybe leaked out. So all kinds of possibilities are there, so you cannot completely guarantee that cryptography is going to keep the data valid and data integrity intact, but you know if you are using strong cryptography, strong hashing function, collision-resistance hashing function for now you are okay. The other issue is that quantum computing.

So in the future if the data is kept for a long time if a particular blockchain is there for a long time and then the keys are still using let us say RSA and even the PKI is RSA, then if quantum computing becomes a reality, RSA will be

broken and then you will have a problem. So, all these things has to be always considered in the context of current time and what is best known about the cryptography you are using at that current time.

(Refer Slide Time: 10:41)



So, another one I heard is that from the government agencies that government must design a blockchain on its own and force every government organization to use that implementation of the blockchain. So, this actually was told to me by some government people saying that you know we want full control on the blockchain that people are using for e-governance, we do not want a foreign provenance blockchain, we do not want a freeware or open-source blockchain, we want to have our own and therefore we will have to force everybody to use it.

One example they were trying to give in support of this argument is that look at what WhatsApp has done to us, we have no way to force Facebook to give us the messages that are being exchanged on WhatsApp. So we should not be dependent on technology built by others, but this is a very wrong idea, the reason being that you cannot have an untrustworthy system to create people's trust. The whole point of doing blockchain based e-governance is to actually earn people's trust in a much better manner.

Now if you say that I am going to build a blockchain, I am going to build a cryptography in it, I am going to have a full control of the code and I want security by obscurity, then you are going to be in for a surprise. First of all, people might suspect that the cryptography has been weakened or some form of some backdoor is there in the infrastructure for government to actually get the information before cryptography is applied to some data or before hashing is done on data that is the one problem with respect to generating citizen trusts, but the other issue is that indeed it is not something that government should control.

(Refer Slide Time: 13:20)

Prevailing misconceptions #5

- One has to have a government vetted closed source blockchain platform on which to develop e-governance solutions

So same kind of misconception is that it has to be a government vetted closed source blockchain platform to develop all e-governance solutions, same thing, but the other problem is that there are multiple different varieties of blockchain right. So we have seen blockchain as evolved from blockchain 1, blockchain 2.0, blockchain 3.0 and now we are talking about blockchain 4.0. So this thing is an evolving technology.

If you make it a closed source blockchain now today, then tomorrow the blockchain will move to much better features and better ways of doing, building applications and then it will be missing out. Second problem is where will you get the best programmers that are working on Ethereum or Hyperledger because these are open sourced and huge community of very good programmers are developing them. Where will you get that kind of developers in the government sector to build something that competes in performance, competes in feature set, competes in the ability to provide guidance, everything is going to be difficult.

So it is best to do open source blockchain platform, because it is open source you can have the full you know authority and ability to actually look into the code, look for existence of any backdoor, you know check the cryptography implementation everything, right. So it is not a good idea to have a government vetted closed source blockchain platform for e-governance solutions.

(Refer Slide Time: 15:09)

Prevailing misconceptions #6

- Blockchain cannot have data-privacy

Another misconception I keep hearing is that oh blockchain is an open ledger, it is open to everybody, so there will be no data privacy and that is again a wrong idea. First of all, you do not have to put the entire data on the blockchain even if it is open and public. You can put the hashes of the data because eventually you want to use blockchain to provide data integrity guarantees and not to float the data itself. In case you want to have the data transferred, there may be other ways to have the data transfer rather than using the blockchain as a conduit for data transfer.

So it is a total misconception or maybe the mindset that is stuck in blockchain 1.0 like bitcoin or Ethereum, where all the data is available to everybody in their public permissionless setup, then I can understand that misconception has arisen of that, but you know people should study the latest blockchain technologies, the distributed ledger technologies and get rid of this misconception.

(Refer Slide Time: 16:36)

Prevailing misconceptions #7

- "Right to forget" in the data privacy law (being tabled in the parliament) is incompatible with blockchain usage

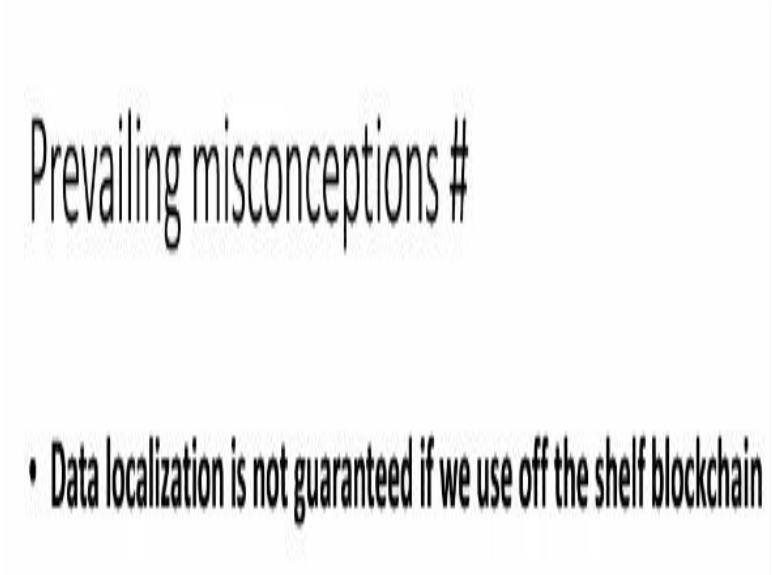
So the other misconception is that the right to forget in data privacy law is incompatible with blockchain usage. So,

this is again the same kind of mistake. First of all, you do not put the data into the blockchain if there are personally identifying information or other kind of information that comes under the purview of the privacy law. So secondly you know the data in a permissioned blockchain, a particular change in the data is possible by basically making a transaction for doing that. So, therefore 2 things.

One is that you do not put e-governance blockchain in public permissionless setup, so therefore there is only permissioned entities that are going to look at the data, but even if that is a concern, you actually can put encrypted version of the data and this encryption has to be strong and forward secure which means that it should be encrypted with the quantum resistant cryptography and then there may be a proxy re-encryption scheme and other schemes to actually for the data principle to release the right to look at the data.

Then once your data has to be forgotten, then you actually destroy the ability to create re-encryption keys. So also in some cases, you only put the hash of the data in which case there is no concern about this right to forget because there is no way to reconstruct your data from the hash. So, again this is a mindset that is coming from knowing only about bitcoin and maybe Ethereum, but this mindset has to change because now you know we have come a long way from bitcoin type of blockchain to distributed ledger technology and where this kind of concerns can be easily addressed.

(Refer Slide Time: 19:00)



Another issue is that oh so if you put something in the blockchain, you do not know the ledger is replicated everywhere, so there is no guarantee that the data will be localized in the country, whereas government requires that all the data should be localized in the country which means that all the data should not leave the country's geographical boundaries. All servers in which data is kept should be within the geographical boundaries of the country because the data may be subpoenaed by the courts of the country.

If it is in another country, it cannot be subpoenaed by the extent laws of the nation. All these considerations are called data localization considerations. Now if you are using a Hyperledger instance or Corda instance or some permissioned private blockchain for providing e-governance with the proper APIs for people to access data, but through certain nodes that are exposing the rest APIs, then all the servers and everything are under your control under the control of the government.

So, there is no concern about data localization. So, this is again another misconception that has to understand how the permissioned private distributed ledger technology works and then this misconception will be addressed.

(Refer Slide Time: 20:47)

Another misconception I heard a lot is that the government must have a regulation to regulate the technology. Now, that is actually again a misconception. You do not regulate a technology, what you regulate is the rules, what you regulate is how the data is used or how transactions take place. So, it is not the underlying technology that you need to regulate, you need to regulate the particular application domain in which if it is banking then banking regulations, if it is legal contracts then it is the company law or whatever be applicable laws are has to regulate that particular application.

Now, one situation is certainly important is that for example our transactions that is done over an IT system without any manual intervention, legal transactions or contract made on an IT system without manual intervention are acceptable or are proof of temper that one can generate from the blockchain, the investigation into a blockchain is that acceptable in a court of law in the country? These kind of blockchain related changes in IT law or IT Act associated with taking into account that the usage of blockchain at various sectors has to be made.

But that is not about regulating the blockchain technology is about, making laws more amenable for broader use of blockchain technology, so that is a facilitator of developing applications on blockchain technology other than regulating the blockchain technology. Regulating blockchain technology would mean that you go and tell that you cannot use this kind of blockchain technology in this kind of applications, you cannot use permissioned blockchain for this kind of applications or you have to use permissioned blockchain technology for this kind of application.

Those would be regulating the technology but that should be less to the particular domain and particular may be government agency would be using this and assuming that they are doing the right thing. There should be certain applicable policies related to, security related to, privacy related to availability, integrity, etc., and they have to choose the right blockchain technology for fulfilling those obligations. You do not have to actually regulate the blockchain technology.

(Refer Slide Time: 23:59)

Then another prevailing misconception I heard a lot is that oh it is all about cryptocurrency, it is immensely power consuming and therefore it is not environment friendly. Again these kinds of ideas are coming from the misunderstanding and the ideas that originated in bitcoin or blockchain 1.0 with proof-of-work. We have seen throughout these classes that is not the only type of blockchain, neither are they really applicable to most of the applications that we really want blockchain to be used in and therefore this concern is not a correct or valid concern.

So with that, I would like to close this week as well as the course. I am hoping that this course has given you a very good conceptual understanding of the blockchain technology, its evolution from blockchain 1.0 to 3.0 and the various ways the blockchain has evolved for making it applicable for various applications, what a blockchain technology

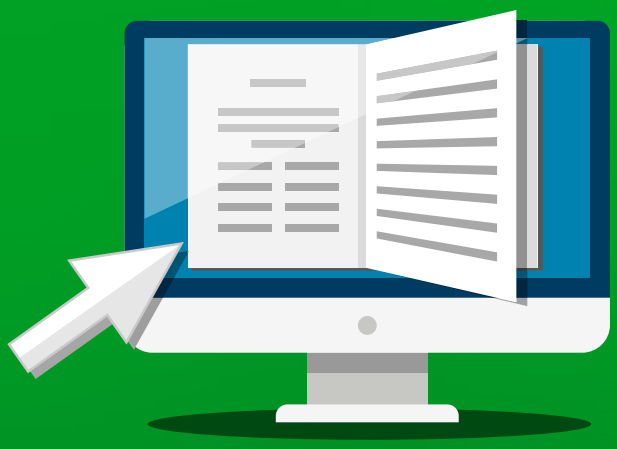
based application give you and what are the advantages of using that particular application over a blockchain rather than a traditional centralized IT system.

Also, I hope that you understand the prevailing misconceptions regarding blockchain and how to dispel them, and finally I hope that by understanding at least 4-5 different blockchains in some details like bitcoin, Ethereum, Hyperledger, Corda, IOTA and having very cursorily seen something like that is the Algorand kind of blockchain, you have some idea that this field of blockchain technology has quite a bit of variety and because of this variety, it is able to fit into various different application domains.

Which application domain actually requires a blockchain based solution depends on what are its requirements, in terms of data integrity, in terms of availability, in terms of you know redundancy of data, in terms of who is allowed to participate, and in terms of whether the data that is kept in this decentralized ledger should be visible to everybody or not, all that considerations has to be factored in before you choose whether to first of all first you choose whether to really use blockchain or centralized system is sufficient for your requirements or if it is the blockchain that you want to do.

Then the question is what type of blockchain you need to use. So, these kind of perceptual understanding is what the goal of this course was and I hope that having gone through 8 weeks of lectures and quizzes and all that you are now comfortable and confident about that kind of abilities to advise as well as select yourself the right approach to developing an application with blockchain. With that, I wish my best to all of you for a more productive career in using blockchain technology in case you decide to do so.

For that also you have to learn more about at least one or two specific blockchain technology in detail how to build applications in those blockchain technology and how to deploy them, how to may be deploy them on the cloud, these kind of things that you have to figure out over time and hopefully this course was a suitable substrate on which you will be building a much more robust castle of knowledge, which will give you advancement in your career and your studies and research whatever is you are doing. With that, thank you all for listening for 8 long weeks and wish you all the best. Thank you.



THIS BOOK IS NOT FOR
SALE NOR COMMERCIAL USE