Introduction to Blockchain Technology & Applications

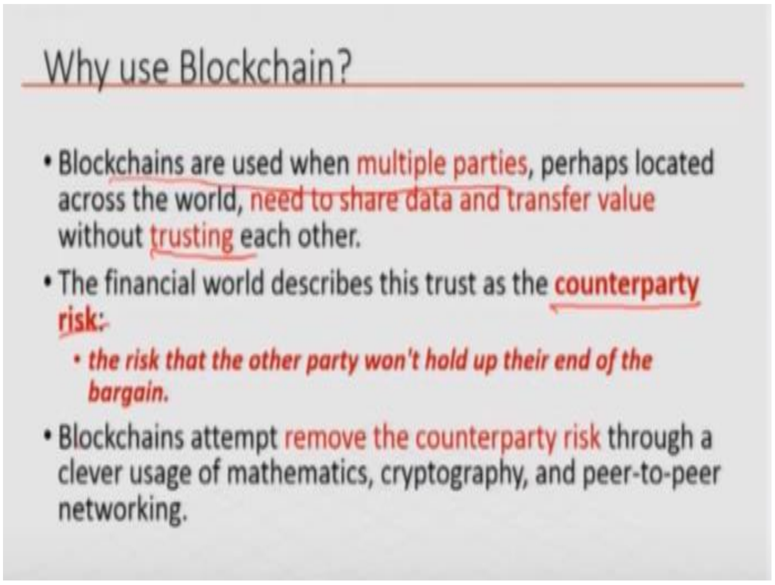
Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

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 it 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)



Why use Blockchain?

- Blockchains are used when multiple parties, perhaps located across the world, need to share data and transfer value without trusting each other.
- The financial world describes this trust as the **counterparty risk**:
  - ***the risk that the other party won't hold up their end of the bargain.***
- Blockchains attempt remove the counterparty risk through a clever usage of mathematics, cryptography, and peer-to-peer networking.

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.

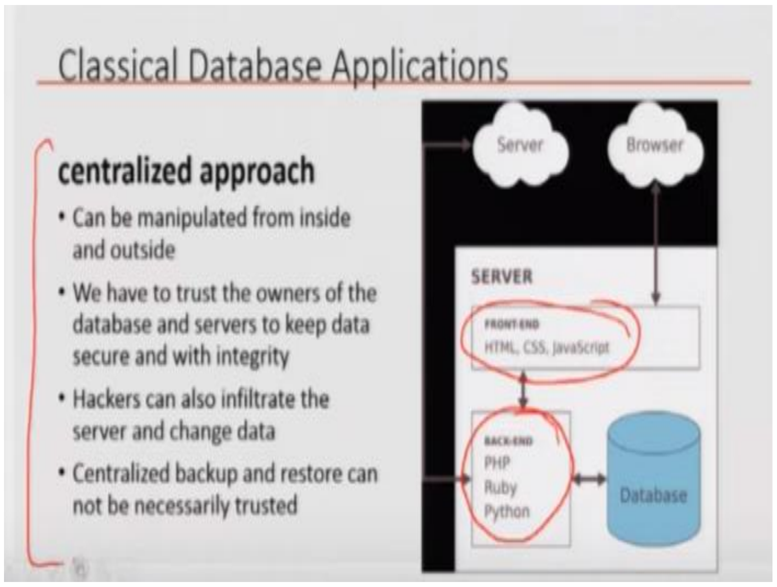
So usually we need a third party. Now this trust about the different parties in a transaction and when I say transaction here I do not mean cryptocurrency transaction any world real world transaction like buying a property or sharing information and all that. So there is a, what the financial world calls this kind of mistrust is counterparty risk. Counterparty risk means that the thought that 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)


Classical Database Applications

centralized approach

- Can be manipulated from inside and outside
- We have to trust the owners of the database and servers to keep data secure and with integrity
- Hackers can also infiltrate the server and change data
- Centralized backup and restore can not be necessarily trusted

Server
Browser
SERVER
FRONT-END
HTML, CSS, JavaScript
BACK-END
PHP
Ruby
Python
Database

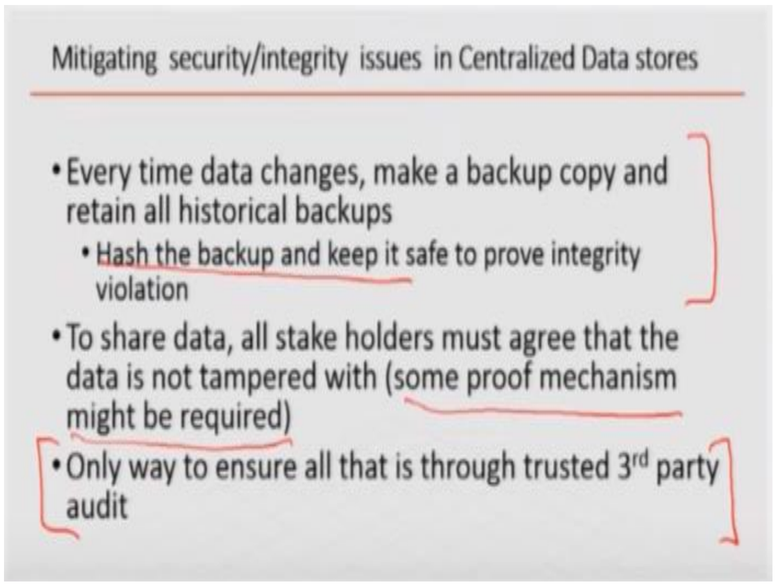
So let us see what is wrong with the centralized approach. And we discussed this before. Let us say the central database is a your student grade database in a university and all the grades submitted by the instructors are in the database. And the way to interact with the database is through front end, like a 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 the 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)



Mitigating security/integrity issues in Centralized Data stores

- Every time data changes, make a backup copy and retain all historical backups
  - Hash the backup and keep it safe to prove integrity violation
- To share data, all stake holders must agree that the data is not tampered with (some proof mechanism might be required)
- Only way to ensure all that is through trusted 3rd party audit

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.

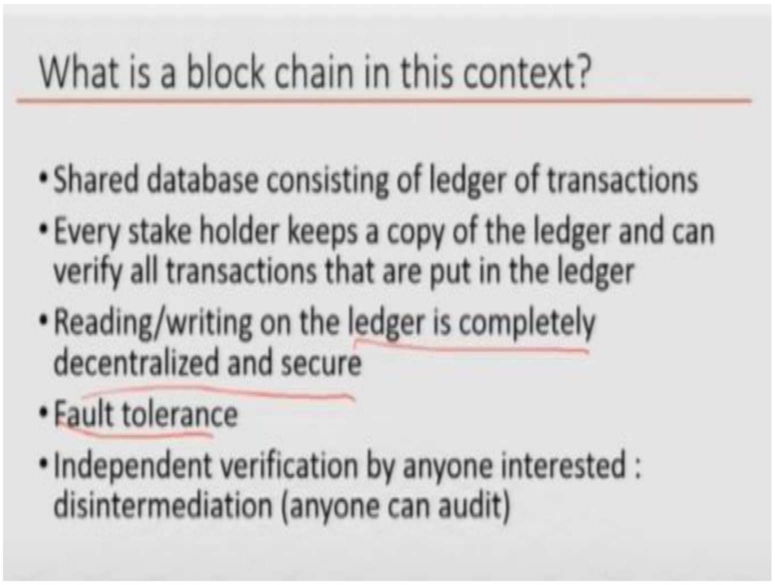
Then later on, if he decides to go and change the data, he will change the hash also. So the hash should be somewhere that he or she cannot manipulate, right. So one possibility is that you do not put all the data in your database in a blockchain, but you commit the hashes of the different backups to 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 have 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)



What is a block chain in this context?

- Shared database consisting of ledger of transactions
- Every stake holder keeps a copy of the ledger and can verify all transactions that are put in the ledger
- Reading/writing on the ledger is completely decentralized and secure
- Fault tolerance
- Independent verification by anyone interested : disintermediation (anyone can audit)

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 deleted 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 to go and browse the blockchain, it is public, and then you can say that, okay you know I have a proof that so 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.

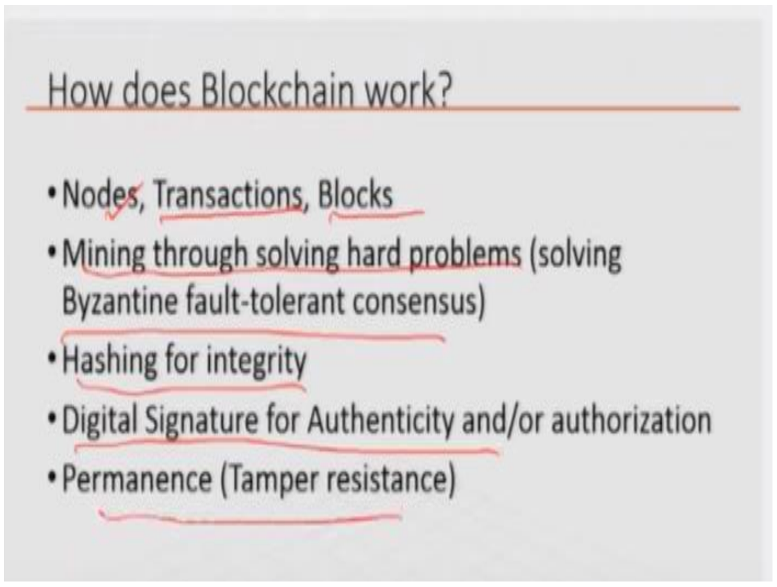
You cannot put grades of students in a public ledger. So in that case, you do not necessarily have to put the actual data in the blockchain, but you put the hash of the data. Every time you make a transaction on the data, you put a hash of the data in the blockchain. So every time you want to check data 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)



How does Blockchain work?

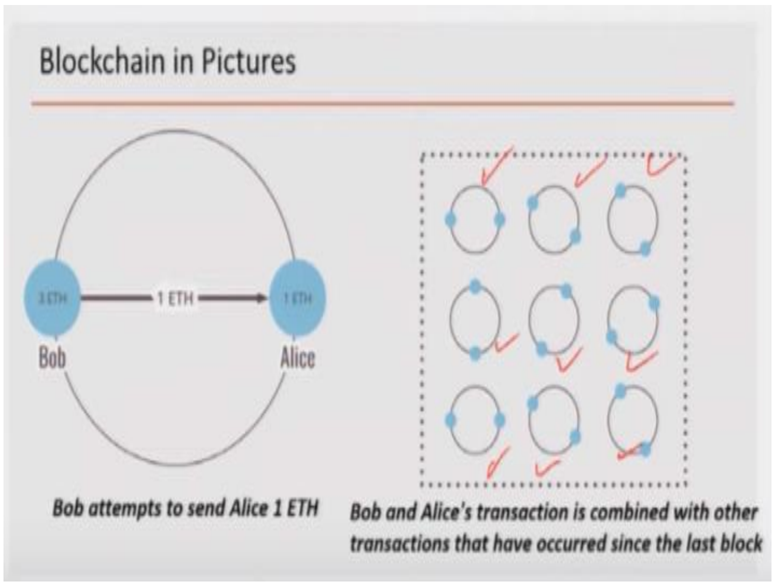
- Nodes, Transactions, Blocks
- Mining through solving hard problems (solving Byzantine fault-tolerant consensus)
- Hashing for integrity
- Digital Signature for Authenticity and/or authorization
- Permanence (Tamper resistance)

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)



Blockchain in Pictures

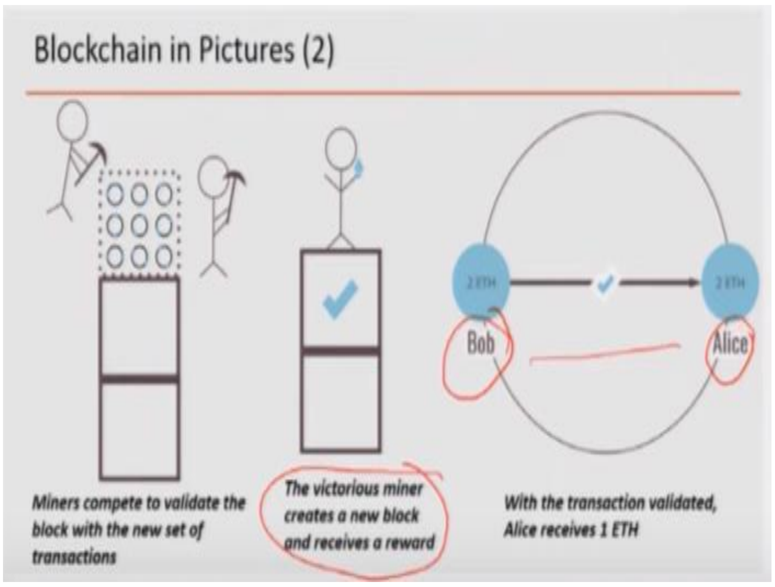
Bob attempts to send Alice 1 ETH

Bob and Alice's transaction is combined with other transactions that have occurred since the last block

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 put attach it to the last block. So that is what happens here.

(Refer Slide Time: 13:24)



Blockchain in Pictures (2)

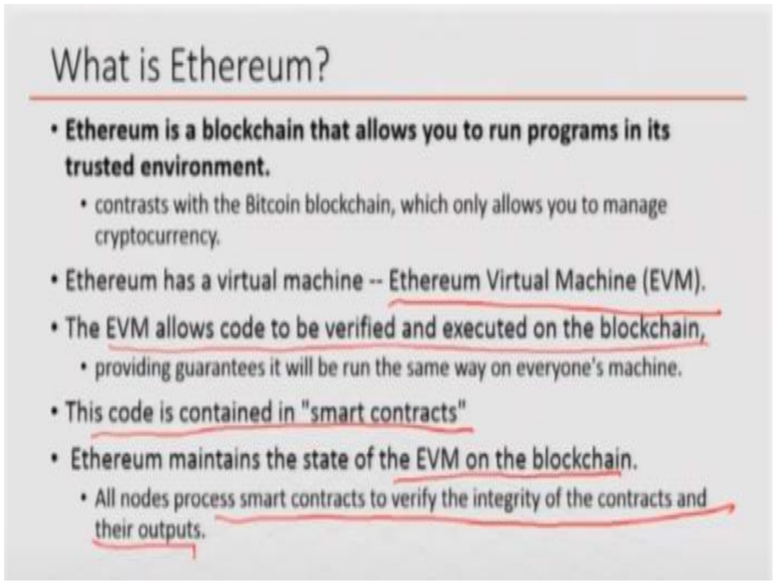
Miners compete to validate the block with the new set of transactions

The victorious miner creates a new block and receives a reward

With the transaction validated, Alice receives 1 ETH

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)



What is Ethereum?

- **Ethereum is a blockchain that allows you to run programs in its trusted environment.**
  - contrasts with the Bitcoin blockchain, which only allows you to manage cryptocurrency.
- Ethereum has a virtual machine -- Ethereum Virtual Machine (EVM).
- The EVM allows code to be verified and executed on the blockchain,
  - providing guarantees it will be run the same way on everyone's machine.
- This code is contained in "smart contracts"
- Ethereum maintains the state of the EVM on the blockchain.
  - All nodes process smart contracts to verify the integrity of the contracts and their outputs.

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, the by 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 the they have a stack and then and they put the data values on the stack and when they actually get to an opcode, they 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 this is how 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 go 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 refers to a smart contract, 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.

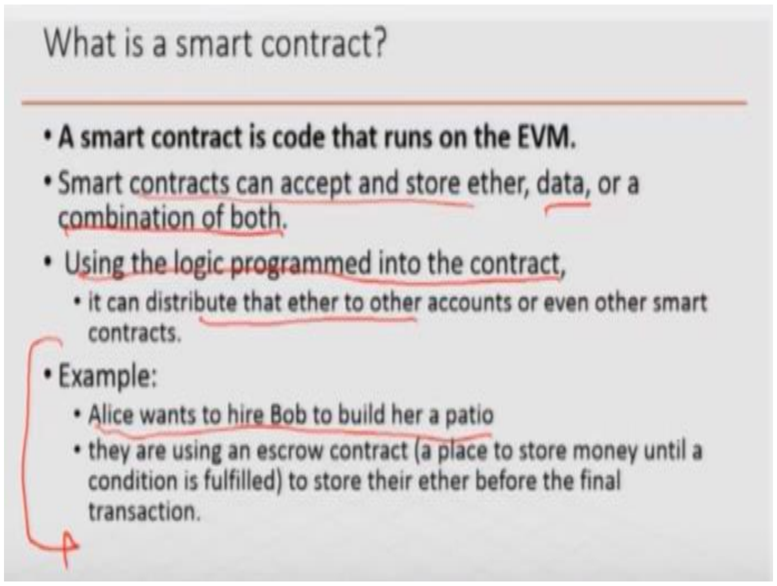
And this programs therefore, have to be what we call deterministic programs. So a deterministic program is a program that no matter how many times you run they will always produce the same result. Unlike a non-deterministic program or a randomized program, where different runs may produce different results.

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)



What is a smart contract?

- **A smart contract is code that runs on the EVM.**
- Smart contracts can accept and store ether, data, or a combination of both.
- Using the logic programmed into the contract,
  - it can distribute that ether to other accounts or even other smart contracts.
- Example:
  - Alice wants to hire Bob to build her a patio
  - they are using an escrow contract (a place to store money until a condition is fulfilled) to store their ether before the final transaction.

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.

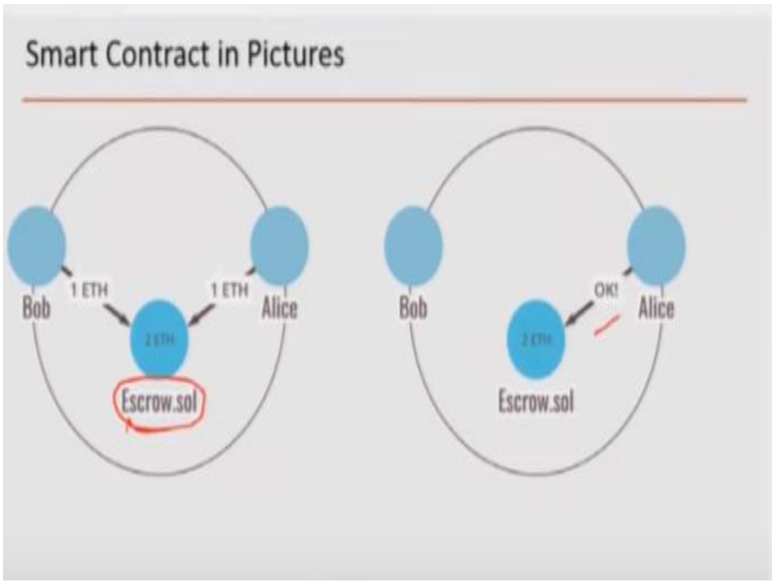
So let us see what kind of transactions are done with smart contract. And here you will not see much different from what this smart contract that we are going to show now is very similar to what can be what we have seen before in terms of escrow transactions, in case of bitcoin. Because this is this 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. And so 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.

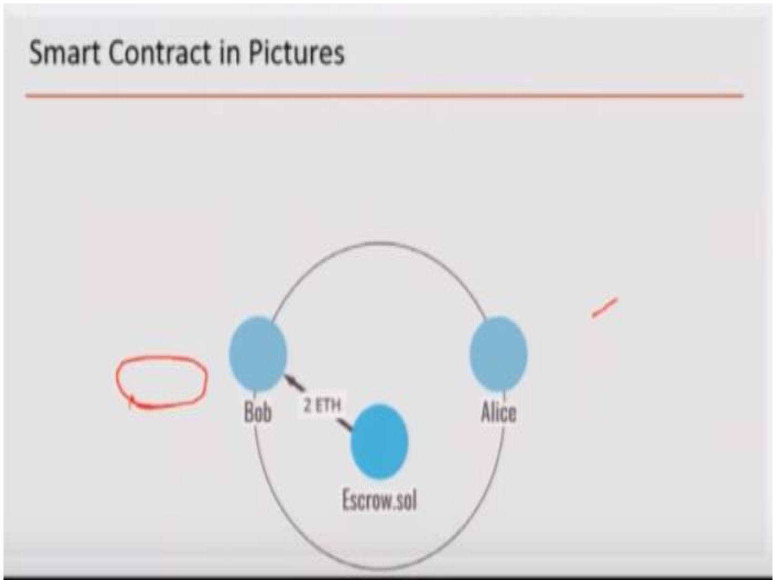
(Refer Slide Time: 25:14)



Smart Contract in Pictures

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.

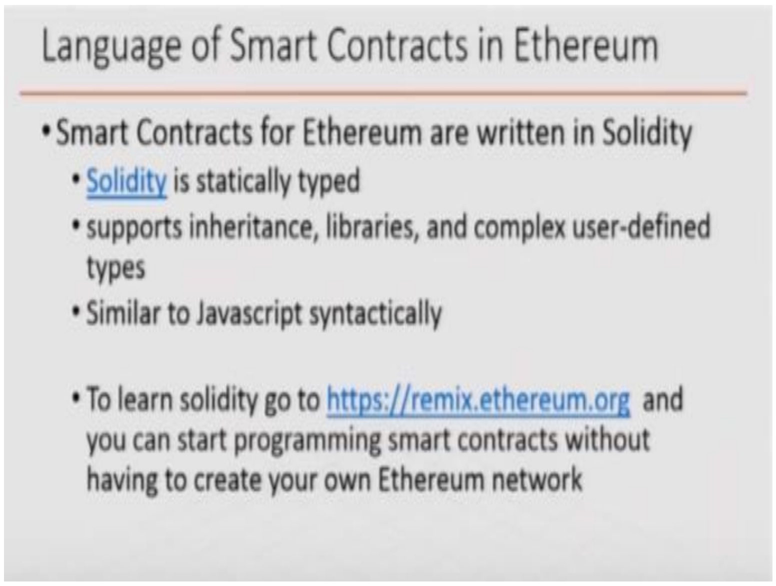
(Refer Slide Time: 25:33)



Smart Contract in Pictures

So one, his deposit back, and one for what he has earned.

(Refer Slide Time: 25:39)



Language of Smart Contracts in Ethereum

- Smart Contracts for Ethereum are written in Solidity
  - Solidity is statically typed
  - supports inheritance, libraries, and complex user-defined types
  - Similar to Javascript syntactically
  - To learn solidity go to https://remix.ethereum.org and you can start programming smart contracts without having to create your own Ethereum network

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.