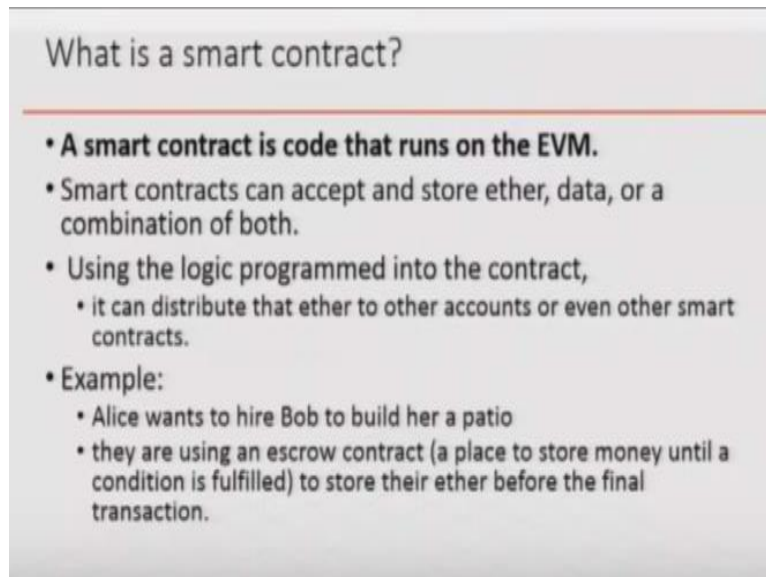


**Introduction to Blockchain Technology & Applications**  
**Prof. Sandeep Shukla**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology-Kanpur**

**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)**



What is a smart contract?

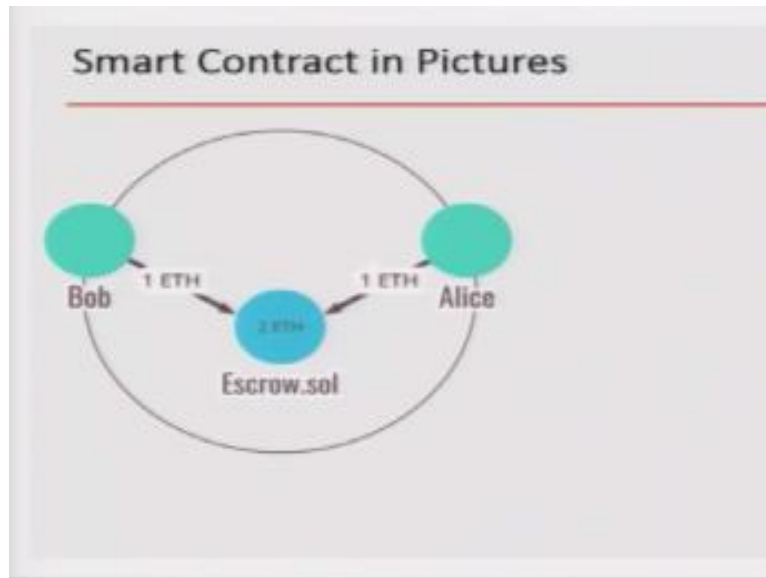
- **A smart contract is code that runs on the EVM.**
- Smart contracts can accept and store ether, data, or a combination of both.
- Using the logic programmed into the contract,
  - it can distribute that ether to other accounts or even other smart contracts.
- **Example:**
  - Alice wants to hire Bob 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 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 also 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.

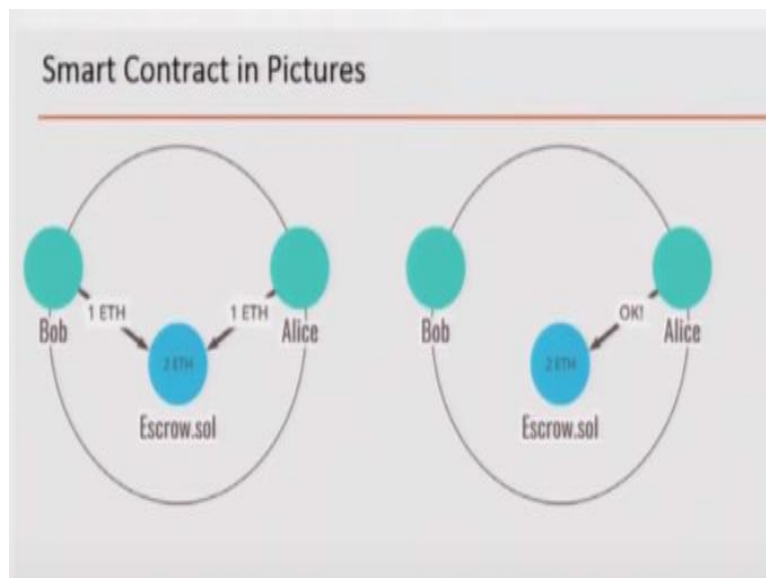
And the smart contract is programmed to accept a command from Alice that says that I am satisfied and now you can pay.

**(Refer Slide Time: 01:56)**



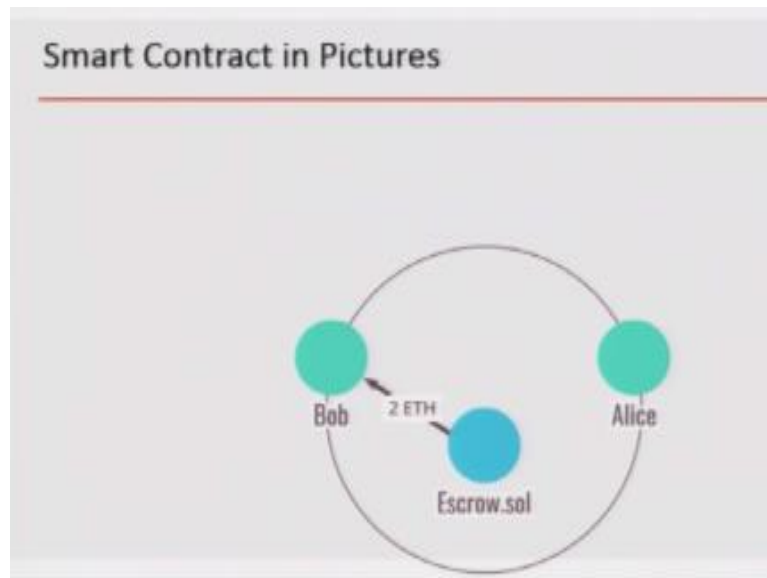
So pictorially this is what it looks like. So Escrow.sol is a solidity program which is a smart contract which will be deployed on Ethereum and Bob, basically and Escrow.sol has functioned to accept ether. So Bob will present 1 ether and Alice will send 1 ether. So now we will have 2 ethers in the escrow.

**(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)**

### 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

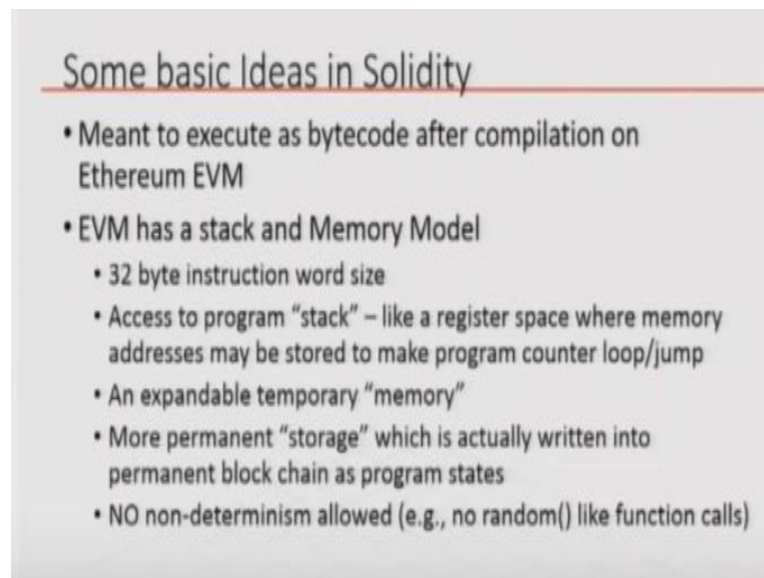
Now the language of smart contract in cases of Ethereum is called Solidity. And so Solidity is kind of like an object oriented language, and it supports inheritance. So you can inherit from given contract, you can inherit and enhance the contract. It has libraries and complex user-defined types. And 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 where you can it is [remix.ethereum.org](https://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)**

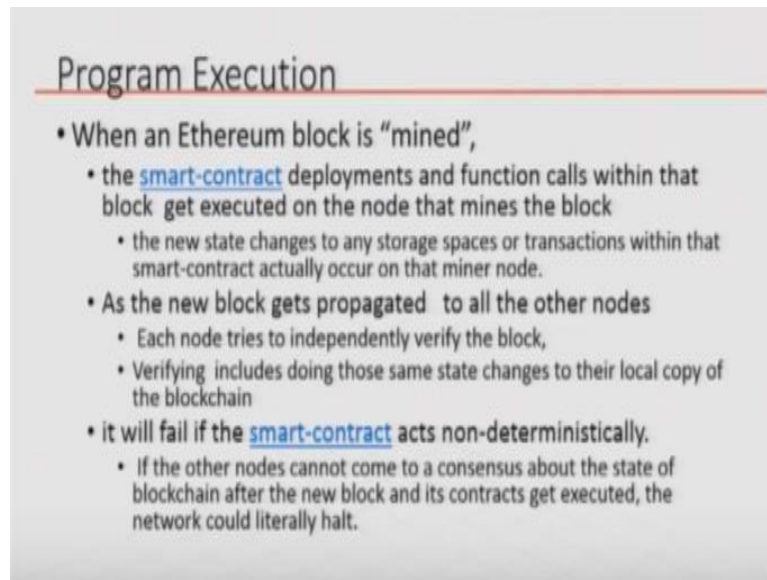


So some basic idea about Solidity. So Solidity when it is compiled, it becomes bytecode, which is going to run on Ethereum Virtual Machine or EVM. And the EVM has a stack based memory, stack based execution model. So it has 32 byte instruction word size, so 32 times 8. That is about 250. It is exactly 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)**



Program Execution

- When an Ethereum block is “mined”,
  - the [smart-contract](#) deployments and function calls within that block get executed on the node that mines the block
    - the new state changes to any storage spaces or transactions within that smart-contract actually occur on that miner node.
  - As the new block gets propagated to all the other nodes
    - Each node tries to independently verify the block,
    - Verifying includes doing those same state changes to their local copy of the blockchain
  - it will fail if the [smart-contract](#) acts non-deterministically.
    - If the other nodes cannot come to a consensus about the state of blockchain after the new block and its contracts get executed, the network could literally halt.

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.

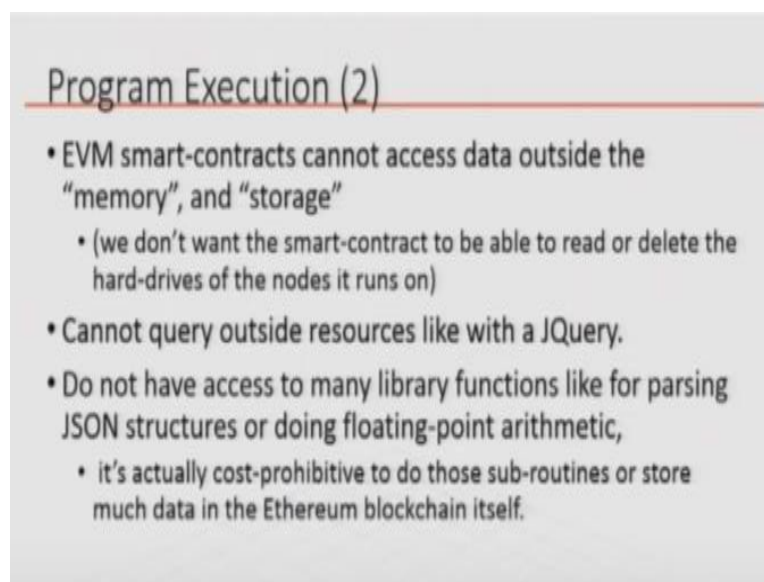
And all the transactions, these things are recorded in the block that it creates. Now once the block is created in a proof of work environment, it has to also solve the hash puzzle. So multiple nodes might be trying to do the mining at the same time, but whoever solves the hash puzzle will be actually 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)**



The slide is titled "Program Execution (2)" and contains a list of constraints on EVM smart-contracts. The text is as follows:

- EVM smart-contracts cannot access data outside the "memory", and "storage"
  - (we don't want the smart-contract to be able to read or delete the hard-drives of the nodes it runs on)
- Cannot query outside resources like with a JQuery.
- Do not have access to many library functions like for parsing JSON structures or doing floating-point arithmetic,
  - it's actually cost-prohibitive to do those sub-routines or store much data in the Ethereum blockchain itself.

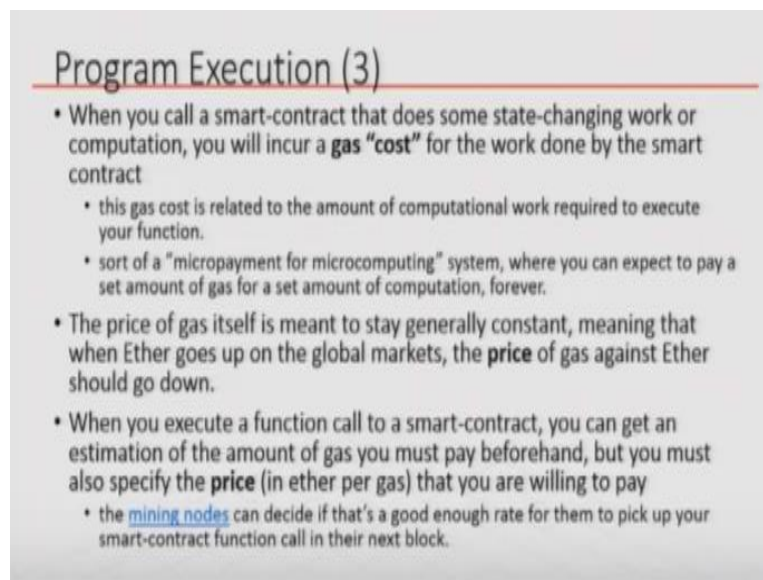
Now EVM smart contracts cannot access data that is outside its memory or storage. So it would not be able to read information from hard drive or other resources on the machine. And it would not be able to also delete or do anything. Because then if that was possible, then there will be no the smart contracts 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)**



Program Execution (3)

- When you call a smart-contract that does some state-changing work or computation, you will incur a **gas "cost"** for the work done by the smart contract
  - this gas cost is related to the amount of computational work required to execute your function.
  - sort of a "micropayment for microcomputing" system, where you can expect to pay a set amount of gas for a set amount of computation, forever.
- The price of gas itself is meant to stay generally constant, meaning that when Ether goes up on the global markets, the **price** of gas against Ether should go down.
- When you execute a function call to a smart-contract, you can get an estimation of the amount of gas you must pay beforehand, but you must also specify the **price** (in ether per gas) that you are willing to pay
  - the [mining nodes](#) can decide if that's a good enough rate for them to pick up your smart-contract function call in their next block.

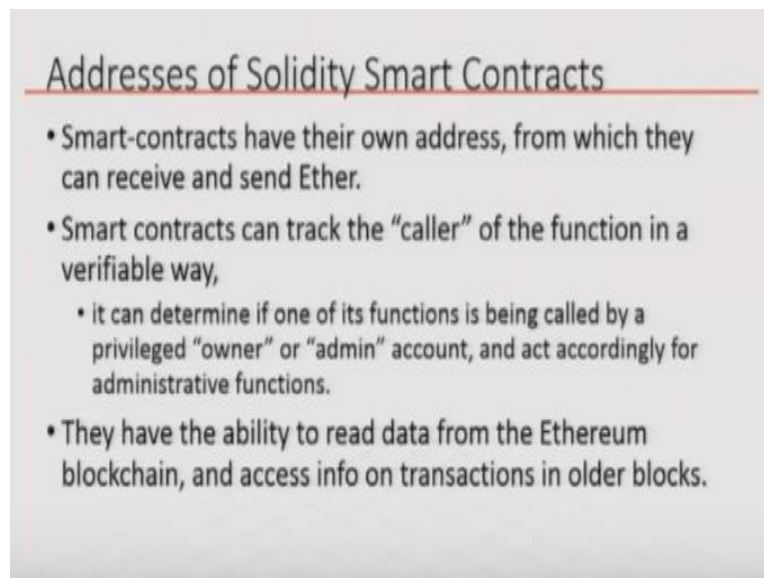
So when you call a smart contract, and it does state changing work or computation, you incur gas cost as I said, and the gas cost is related to the amount of computational work that is required to execute the function. And so you can think of this as kind of like as micro payment per micro computing 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.

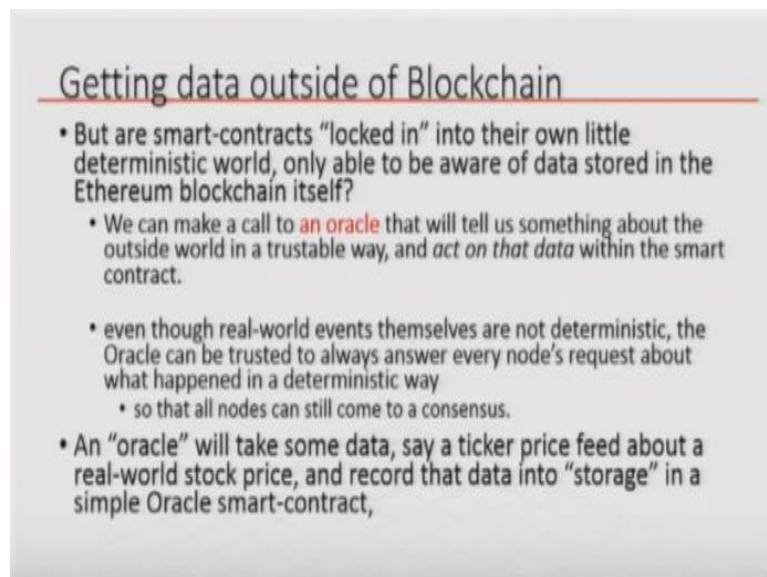


But then also there are smart contract accounts, which are actually self-managed accounts and so they have their own addresses, like any other Ethereum account address. And the smart contracts can also track who is calling the contract.

And therefore if the contract caller has a, 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 was 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)**



Getting data outside of Blockchain

- But are smart-contracts “locked in” into their own little deterministic world, only able to be aware of data stored in the Ethereum blockchain itself?
  - We can make a call to an **oracle** that will tell us something about the outside world in a trustable way, and *act on that data* within the smart contract.
  - even though real-world events themselves are not deterministic, the Oracle can be trusted to always answer every node’s request about what happened in a deterministic way
    - so that all nodes can still come to a consensus.
- An “oracle” will take some data, say a ticker price feed about a real-world stock price, and record that data into “storage” in a simple Oracle smart-contract,

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 the 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 there is a 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)**



The slide is titled "Ethereum networks" and contains three bullet points. The first bullet point mentions "MainNet" in red text. The second bullet point mentions "US Dollars".

- On the **MainNet**, data on the chain—including account balances and transactions—are public, and anyone can create a node and begin verifying transactions.
- Ether on this network has a market value and can be exchanged for other cryptocurrency or fiat currencies like US Dollars.
- But there are other Ethereum networks as well.

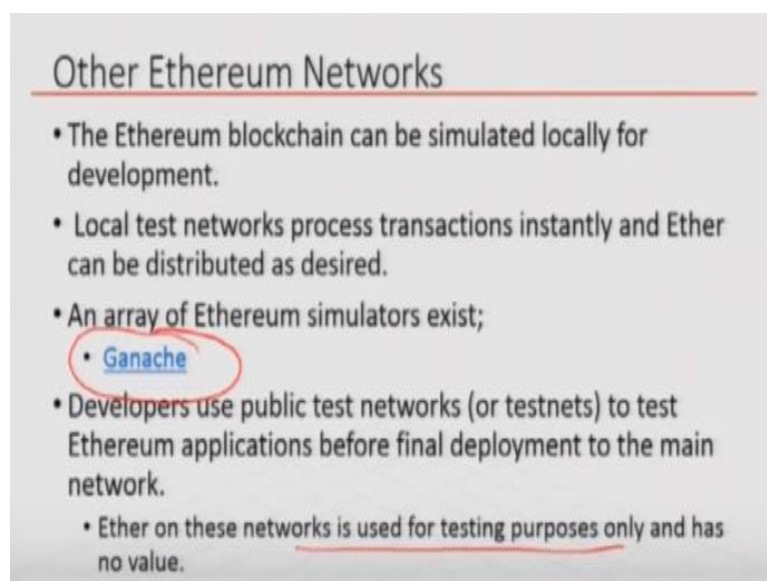
So Ethereum has multiple different network. So blockchain for example, has a single network, right. So of course, you can create a test network also for blockchain. But Ethereum Foundation actually maintain multiple different networks. So the MainNet, which is the net on which real Ethereum transactions 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)**



Other Ethereum Networks

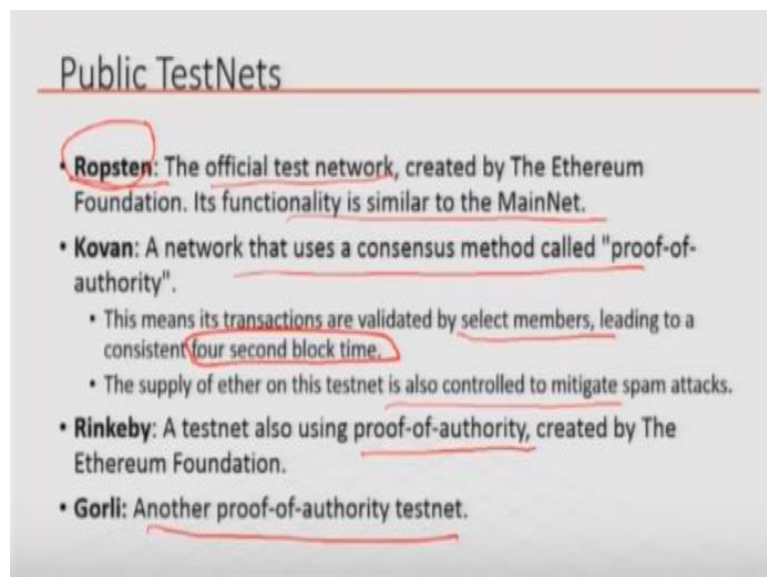
- The Ethereum blockchain can be simulated locally for development.
- Local test networks process transactions instantly and Ether can be distributed as desired.
- An array of Ethereum simulators exist;
  - [Ganache](#)
- Developers use public test networks (or testnets) to test Ethereum applications before final deployment to the main network.
  - Ether on these networks is used for testing purposes only and has no value.

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.

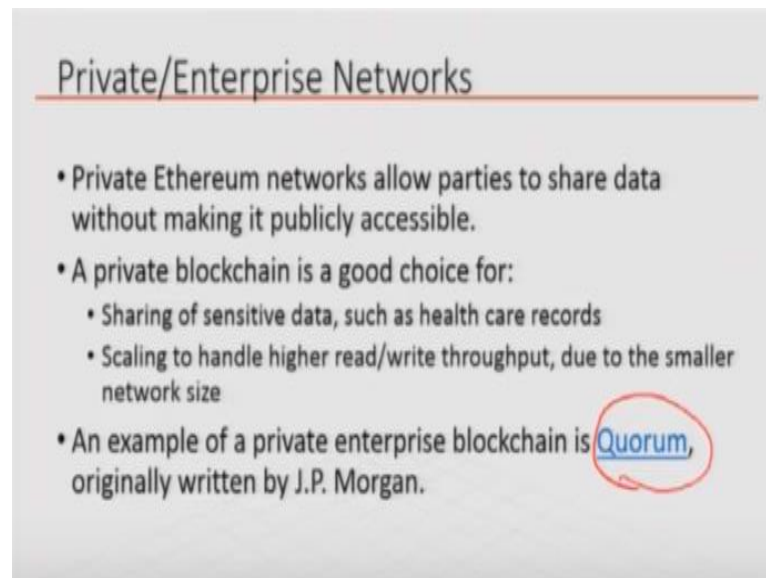
So if they do, if they put as invalid transaction in a block or if they try to approve double spending kind of transactions, you will eventually catch them. So that is a different kind of consensus mechanism. So the proof of work consensus mechanism does not trust anybody. So everybody is equal in the network. And there you actually 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)



The slide is titled "Private/Enterprise Networks" and contains the following bullet points:

- Private Ethereum networks allow parties to share data without making it publicly accessible.
- A private blockchain is a good choice for:
  - Sharing of sensitive data, such as health care records
  - Scaling to handle higher read/write throughput, due to the smaller network size
- An example of a private enterprise blockchain is Quorum, originally written by J.P. Morgan.

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, so 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)**

Distributed Applications (Dapps)

- Applications using smart contracts for their processing are called "distributed applications", or "dapps".
- The user interfaces for these dapps consist of familiar languages such as HTML, CSS, and JavaScript.
- The application itself can be hosted on a traditional web server or on a decentralized file service such as [Swarm](#) or [IPFS](#).
- Dapps based solution available for:
  - Record keeping
  - Finance
  - Supply chains
  - Real estate
  - Marketplaces

*Interplanetary file System*

People now 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 contracts will be 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.