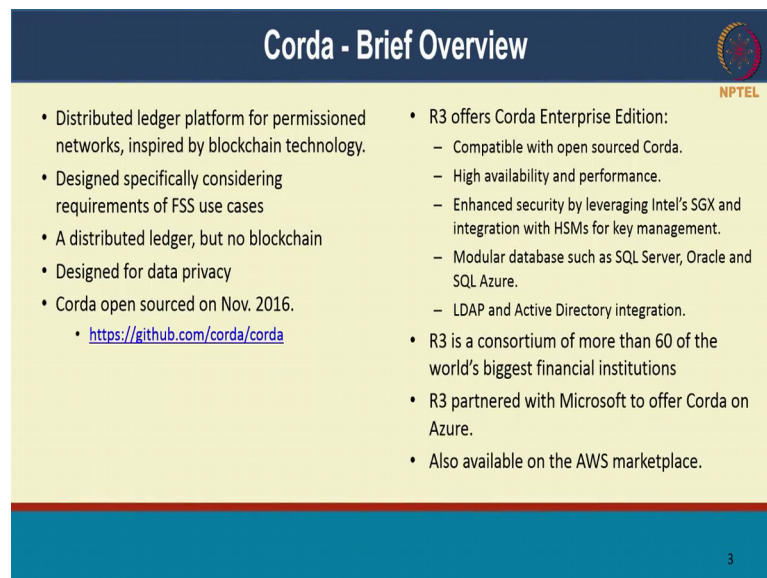


**Blockchains Architecture, Design and Use Cases**  
**Prof. Sandip Chakraborty**  
**Prof. Praveen Jayachandran**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**IBM Research, India**


**Lecture – 28**  
**Comparing Ecosystems**

Hello, everyone. Welcome back to our Blockchains course. We are now looking at different blockchain platforms. This lecture and the next one is going to be we are going to be discussing Corda. This is the first part of Corda.

(Refer Slide Time: 00:27)



**Corda - Brief Overview**

  
NPTEL

- Distributed ledger platform for permissioned networks, inspired by blockchain technology.
- Designed specifically considering requirements of FSS use cases
- A distributed ledger, but no blockchain
- Designed for data privacy
- Corda open sourced on Nov. 2016.
  - <https://github.com/corda/corda>
- R3 offers Corda Enterprise Edition:
  - Compatible with open sourced Corda.
  - High availability and performance.
  - Enhanced security by leveraging Intel's SGX and integration with HSMS for key management.
  - Modular database such as SQL Server, Oracle and SQL Azure.
  - LDAP and Active Directory integration.
- R3 is a consortium of more than 60 of the world's biggest financial institutions
- R3 partnered with Microsoft to offer Corda on Azure.
- Also available on the AWS marketplace.

3

Corda is really a permissioned blockchain network or I should not really say it is a blockchain because I should call it a distributed ledger, but there is not a blockchain in Corda. For instance, no there is not a notion of fully ordered transactions that are all hash chained together. So, there is no really a blockchain as such, but it is a distributed ledger with consensus. So, we will talk about how that works.

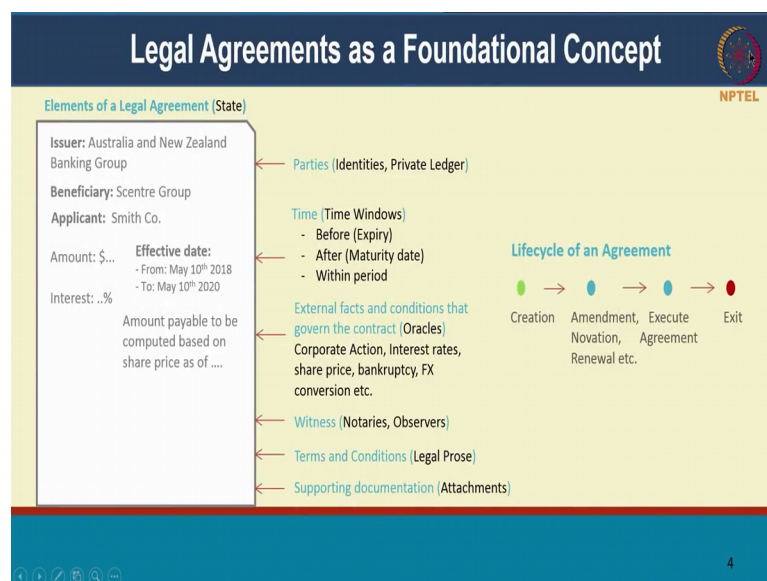
And Corda has been designed specifically considering the requirements of financial services use cases. So, the now bunch of banks have come together to form a consortium called R3 and R3 leads the development of Corda and also the governance around the Corda development itself right. It is specifically designed considering data privacy as a

first class requirement and Corda was open sourced in November, 2016 and has been seeing a good amount of progress in terms of capabilities over the last year and a half.

There is R3 is coming out with an enterprise version of Corda and it is compatible with the open source version with some improvements, right. So, they are promising higher availability and performance, enhanced security using Intel SGX and the notion of enclaves for secure execution and also integration with the hardware security modules for key management. They are also bringing in a modular database. So, we can use different relational databases along with Corda and it also brings an integration with LDAP and active directory.

R3, the consortium of financial institutions has partnered with Microsoft to offer Corda on Microsoft Azure the on cloud. It is also available on AWS market place. So, it is now available for you to instantiate on a public cloud and try it out.

(Refer Slide Time: 02:32)



So, let us look at some of the concepts, right. So, I am going to talk about some of the higher level concept and I will get into some of the details after that technical details itself. So, as a foundational concept R3 is founders and encoders creators they felt that with financial services use cases legal agreements are a critical part of those use cases. So, this could be agreements between banks on how they are going to operate on assumption of liability and risk. It could be regulatory a give agreements maybe between a regulator and a bank. So, they want to associate smart contracts with legal agreements.

So, this is a fundamentally philosophically different notion from Ethereum like we discussed in the previous lectures. Ethereum as I mentioned considered code is the law, right. But, in Corda they consider the code and the legal agreement as to distinct aspects the code is something that tries to support what is mentioned in the legal agreement, but is not distinct from the legal agreement. So, the legal agreement is what is going to be you know we acceptable in a court of law, but the code the smart contract code and the distributed ledger information can support a legal agreement, but it is not really legally admissible.

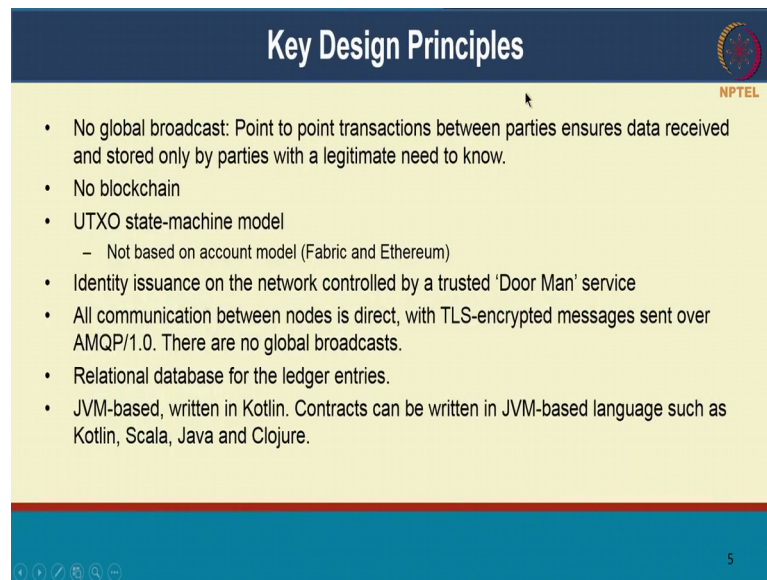
So, elements of a legal agreement; so, this you can think of it as it is going to be captured as a document it can be stored together with the smart contract. So, there is a they that could be this is the optional, but typically there is a legal agreement that goes together with a with a smart contract, whether it is a bond issuance or anything else right that you might see you know or the stock market exchange. Any of those it is there are legal agreements that are associated with the smart contracts.

So, the parties of the agreement are specified. The time window is for when the contract is valid. So, what is it is valid before a particular expiry date or until a particular maturity date maybe there is an agreement for let us say a term deposit so, there is a maturity date for it or a or a duration a period of time. There could be external facts and conditions that govern the contract and these facts may be provided by oracles. Oracles are also a first class entity in Corda. So, we will talk about that in the next lecture not this one.

And, these external facts can be various right it could be particular interest rates that are fixed, it could be a share price that changes over time, but it is it is fixed for all users at any given point of time, it could be foreign exchange conversion there is many different notions of external facts that might influence a contract. And, these external facts are provided to the contract through trusted oracles that could be witnesses. There are terms and conditions of the contract these this could be in legal prose and there could always be supporting documents and attachments along with the legal agreement.

And, you can actually track the lifecycle of the agreement itself from creation to amendment to actual execution of the agreement to an exit, right, where the agreement is no longer valid, right. So, all of this is then captured as a legal agreement and goes together with a smart contract.

(Refer Slide Time: 06:01)



### Key Design Principles

- No global broadcast: Point to point transactions between parties ensures data received and stored only by parties with a legitimate need to know.
- No blockchain
- UTXO state-machine model
  - Not based on account model (Fabric and Ethereum)
- Identity issuance on the network controlled by a trusted 'Door Man' service
- All communication between nodes is direct, with TLS-encrypted messages sent over AMQP/1.0. There are no global broadcasts.
- Relational database for the ledger entries.
- JVM-based, written in Kotlin. Contracts can be written in JVM-based language such as Kotlin, Scala, Java and Clojure.

5

So, what are some of the key design principles of Corda itself? So, the first very fundamental and a very unique principles of Corda is that it does not have any global broadcast. So, if you look at any other blockchain platform there is typically an exchange of information it could be transactions typically that are spread across the network and all the network participants will get that transaction information. So, that is a global broadcast inherently part of every blockchain platform.

Corda in stock contrast, because they value privacy and they want a build that in as a first class first class construct. There is no global broadcast. So, all the message exchange are peer to peer. So, you can only point to another peer and send a message to them you cannot send a message across to the entire network and that is also fundamental because you do not want to send messages you want to keep messages private and only authorized entities should be recipients of that message.

No blockchain measure mentioned. So, it is going to be a distributed ledger. So, every node will maintain its own ledger and in some sense the global distributed ledger is a union of all these ledgers. So, we will see how that we will talk about how that works. It does use a UTXO state model state machine model and it does have a way to verify that there are you are actually trying to use unspent transactions and in your trying to double spend in any way.

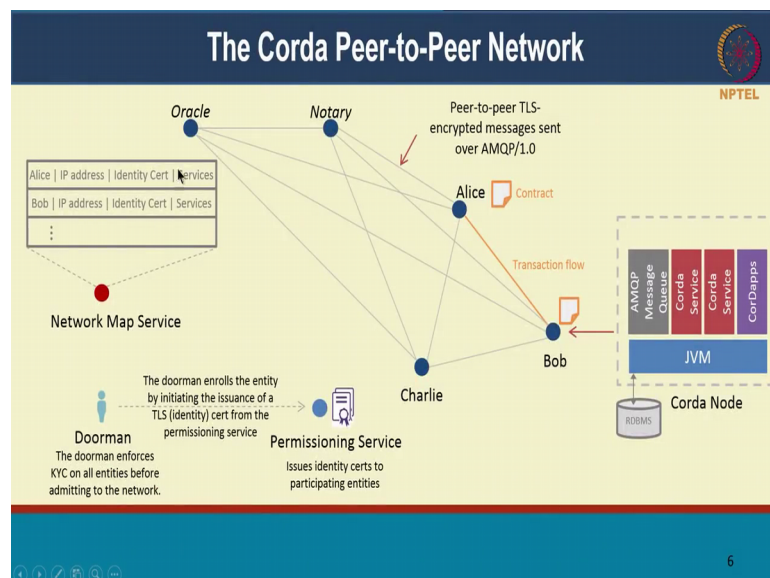


It is not so, it is not an account base model. So, Fabric and Ethereum use an account base model where there are accounts and accounts can store arbitrary state whereas, here there is a UTXO model and you are only going to be transacting with unspent transaction. It is a permission network. So, it is going to have identity issuance on the network that is trusted that is issued by our trusted Door Man service.

We will briefly talk about that as well all communication between nodes is direct as I mentioned it has to be addressed to a particular recipient it is encrypted using TLS. So, it is secure. So, it is a secure message exchange, it is sent over AMQP, right there are no global broadcasts. The ledger itself is a relational database. So, the smart contract can invoke sequel queries to read and write from the database which is the ledger, we will talk about details of how the ledger itself is composed.

And, the smart contract itself the is based on any JVM-based language you can use your contract in any JVM-based language are typically it is Kotlin right some of the contracts they have written are in Kotlin and the virtual machine like Ethereum has the Ethereum virtual machine the contracts run in a JVM based virtual machine. So, it is a java virtual machine on which these contracts run. So, that is the some of the key design principle.

(Refer Slide Time: 09:07)



Corda is of course, a peer to peer network where we can there are multiple entities of this network let us say Alice Bob and Charlie these could be banks it could be financial institutions could be others who are part of this Corda network. There are other entities

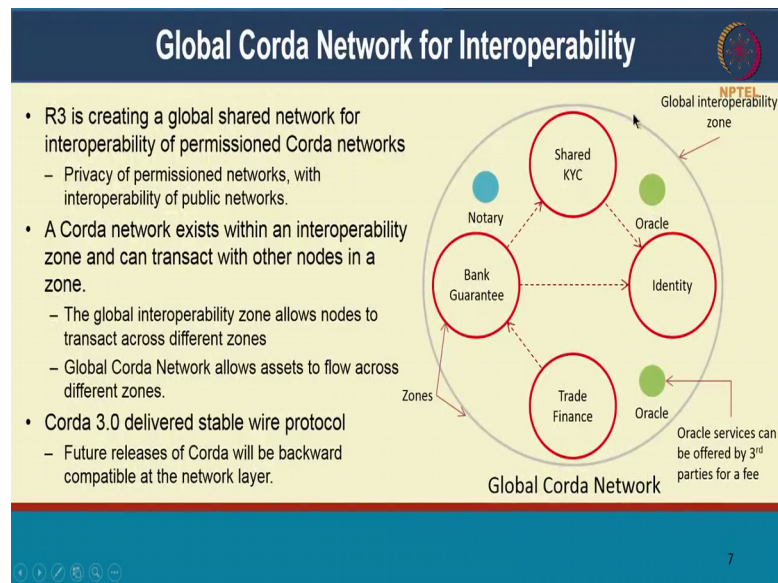
like notaries and oracles. These are in some ways trusted entities, we will talk about the function of notaries and oracles in the next lecture, once we understand how the Corda system the basic set of transactions and contracts work in Corda right.

So, all the message exchanges between peers as well as notaries and oracles all of them happen over TLS-encrypted messages sent over AMQP. There is as I mentioned there is a doorman service. The doorman service is the one that is going to do some form of KYC so, know your customer. They will make sure these are legitimate entities and they require access into this network they need to be part of this Corda network.

And, the doorman once it is approved this entity for access you then issues an identity based on a TLS and this is from a permissioning service. So, it then issues this identity and the permissioning service creates the cert for the participating entities. So, this is how the permissioning itself works. There is a doorman service providing KYC and there is the permission service that issues the certificate itself once the doorman service enrolls it. There is a network map service that maps entities to their peer IP addresses as well as their identity.

So, this is how people will know which entities are part of the system, how do I send a message to another peer, what is the IP address on which I should send it, what is the certificate that I should address it by. So, all that is captured in a network map service that is maintained by the network and anyone can query it. So, the Corda and node itself as I mentioned runs on java virtual machine all the state information is and is stored in an RDBMS, it is a relational database. There is a AMQP messaging service that is how peer to peer message exchanges happen. Then there could be one or more Corda services and then Corda applications right that are running in this in a single node.

(Refer Slide Time: 11:29)



Apart from just a private permissioned network R3 or the Corda community envisions are really a network of networks to evolve. So, there could be for instance a network of entities of working on a shared KYC application there could be another network of entities working on bank guarantees another network for trade finance and so on, right. These could be overlapping networks. So, there could be common entities in this network it is entirely possible.

But, the Corda R3 really envisions all of these blockchain networks, a independent blockchain networks to actually be interacting with one another. They might be leveraging the same oracles for trusted information, they could be using the same notaries for instance, there could be overlap in some of these functions, but apart these all of these networks are connected in a global interoperability zone. So, this global interoperability zone is really a then gets you the notion of a public network.

So, this connection between multiple private networks in a globally shared network gives you both the privacy of permission networks as well as the interoperability across networks. So, I want to do something on KYC and then I want to issue a bank guarantee. So, all of these things can be interoperable and part of a globally shared network.

It allows also each node operates within a particular zone and it can transact with other nodes in that zone, right. So, and so, they the Global Corda Network allows assets to flow across from one zone to another. So, asset can be created in one zone and then it can

be transferred over to another zone. So, those sorts of interoperability functions will become possible with this global interoperability zone.

So, Corda 3.0 delivered a stable protocol for this and the future versions of Corda will be compatible with this. So, this really allows multiple networks to be interoperable and assets to be taken from one network to another network and nodes to really transact across these different zones, ok.

(Refer Slide Time: 13:50)

### UTXO State Machine Model

Previous version of state marked historic (consumed or spent)  
Provides useful audit trail

Transaction

- Shared facts are represented by states using a UTXO state-machine model.
- States are similar to Bitcoin transactions, but can represent arbitrary data.

- States are immutable and represent a shared fact at specific point in time.
- States evolve by allowing new states to replace old states, resulting in a state sequence.

States are statically typed. An IOU state is different from an bond state.

Source <https://bitcoin.org/en/developer-guide>

Now, coming to the transaction model itself, right. So, so Corda uses UTXO model how it works is each time a new transaction is created it has a reference back to a previous transaction that is unspent. And then the previous once a new transaction is created the previous transaction is marked as historic the historic basically is a synonymous with being consumed or that transaction being spent. And, this provides a full audit trail. So, because of these links from one state to another you can have a full audit trail of all of these agreements over a period of time.

So, for instance if there is your maintaining balance in some sense Alice's balance. So, you could have a transaction where Alice pays Bobs Alice borrows money from Bob and then Alice may repay a part of that money. So, all of these are recorded as independent transactions and as a new transaction gets added the previous transaction that it is refers to is marked as spender. This is very similar to the UTXO model except that each of these transactions is going to hold state.

The states apart from unlike bitcoin although this utxo unlike bitcoin, where bitcoin only refers to bitcoin input and bitcoin output here you can really in your state information you can actually have arbitrary data. So, this can really refer to maybe a table in your database and have data pertaining to different rows and columns of that table. So, arbitrary data can be captured as part of state, but as in when state gets modified previous states are linked and they are marked as historic.

So, that way states once they are historic they are immutable and represent a shared fact. So, as in when states get created they are all immutable. So, again it is a append only log of transactions, but it is just that it is not a globally synchronized log it is the all nodes are not maintaining a single blockchain in some sense.

So, this is just a pictorial view of how UTXO works. This is inputs and outputs again, but just that inputs and output can have arbitrary data. Now, coming to the storage aspect itself right each node in the network maintains a vaults, right. It is called a vaults and it is really a database that tracks all the data that that node is privative, right. So, it has all the current and historic states for that node.

(Refer Slide Time: 16:31)

### Vaults and States

- Each node on the network maintains a *vault* - a database where it tracks all the current and historic states that it is aware of.
- Vaults are currently based on the H2 embedded SQL engine. Other databases supporting JDBC are planned.
- The *current state* of the ledger comprises of all unconsumed transactions.

Life-cycle of shared facts B, A, C and S

Corda allows for fine-grained access control at the level of a state-sequence.

In Fabric, data partitioning is managed at the level of Channels and Collections (SideDB).

Contract ref points to a contract which defines the verification function

Participants list the peers who can consume this state in a transaction

**STATE**

<p><b>IOU CONTRACT REF</b></p>	<p><b>IOU STATE PROPERTIES</b></p> <p>From: Alice To: Bob Amount: £10 Due: 01/03/2017 Paid: £5 Penalty: 20%</p>
<p><b>PARTICIPANTS</b></p> <p>Alice Bob</p>	<p>Properties reflect the state of an agreement or contract at a specific point in time</p>

Properties reflect the state of an agreement or contract at a specific point in time

Example of state representing an IOU of £10 from Alice to Bob.

And, different nodes can actually have different view of state information. So, there could be data that is shared between three parties only those three will hold it there could be other data that is hold by just two of them only those two hold it the third party does not hold it and so on.

So, these vaults are currently based on H2 embedded sequel engine and they are also planning to support JDBC, have not looked recently whether they have actually added that, but I believe they can support any relational database. The current state of the ledger comprises of all the unconsumed transactions. So, is very similar to UTXO. So, you can have a sequence of these transactions, the unconsumed state is the current state rest of it is historic state.

Corda allows for fine-grained access control on the state sequence. So, I can say who gets access to which state information I can do that in a fine grained manner. Fabric in contrast allows this data partitioning through the notion of channels and collections. In Corda just the way the message passing works and how the transactions are shared we will see how that works that itself gives you a very strong notion of privacy, right. Only the participants of a transaction will ever see the transaction and the data within it.

So, then how does a how does the distributed ledger work will be the next couple of slides.

(Refer Slide Time: 17:54)

## The Corda Ledger

The ledger is subjective from each peer's perspective and is a union of all facts the peer intersects with (including facts that are not shared). **There is no global broadcast.** The whole ledger is everything that's on-ledger: {1, 7, 6, 5, 9, 4, 3, 2, 8}.

Not all on-ledger facts are shared

Each peer maintains a separate **vault** of facts (think of a fact as a row in a table)

Peer	Fact ID	Fact Description
ALICE	1	"Mach consensus"
	7	"So bilinear"
	11	"How ledger"
BOB	1	"Mach consensus"
	4	"Why fact"
	5	"So bilinear"

Two peers are always guaranteed to see the exact same version of any on-ledger facts they share

Fabric is based on a broadcast architecture. Corda takes a need-to-know approach.

In Corda, there's no trace of private multi-lateral transactions on a "main-chain" (see *Notary* later).

- **No accounts:** Allows for transactions to be applied in parallel.
- **Transaction Ordering:** Transactions ordering is enforced by hash functions that identify previous states.
- **Consensus:** Conflict is a double spend problem.
- **Auditability:** Full history of all activity is recorded.

10

So, the Corda ledger; so, let us look at this right. So, let us say Alice is one node and bob is another node. So, there are certain set of shared facts between Alice and Bob. So, in this case 1 and 7 are shared facts between Alice and Bob. So, both of them will store their, so; store them on their respective ledgers. So, Alice stores 1 and 7. So, the values the facts of them are there. Apart from this Alice might have some state that is not known

to Bob and it will store it on its ledger; likewise Bob has some state that is not known to Alice. So, that is 5 and 6 that is all that store in Bob's ledger, but not on Alice's. So, the vault for each peer is distinct from the other peers and it has only the data that is private to that node or that node is aware of and again there is no global broadcast.

So, no node will know the universal or the full superset of all of this information. You want to look at it as the global state of the network itself you can think of it as the global state of the network or all the facts known by the network is really the union of all the facts known by each entity, right. So, if you put that all together there will be the global knowledge, but no one entity maintains that. So, all facts are split on and shared only by the parties private to that fact or that transaction. So, has no notion of accounts, it is only facts and information that stored amongst parties.

Transaction ordering is enforced by hash functions based on previous states. So, in some sense this is a this is not a total ordering there is only a partially partial ordering between transactions. So, you know that one transaction consumed information from a previous transaction. So, there is a link between those two. So, there is a dependency and an ordering but, there are there could be transactions where there is no ordering right or at least at the level of a single peer they might be parallel transactions.


Consensus allows you identify essentially the double spending problem and determines whether a transaction is valid or not. So, we will discuss consensus and it all because of the fact that we have a full sequence of the provenance of everything where has happened auditability is also achieved, right. You can get a full history of all activity that is recorded on each peer.

So, this is again a fundamental difference fabric Ethereum one of those things they all are based on a fabric and all other permission networks are based on a broadcast architecture when I said Ethereum I meant the permissions to assume of Ethereum. Corda takes a very drastic drastically different view right it is only need to know only pointed communication there is no global blockchain at all.



(Refer Slide Time: 20:42)

## Contracts



- Contracts can be written in any JVM language (Java, Kotlin, Clojure, Scala, etc) and are executed in a *sandbox* (modified version of JVM).
- Contract code must be deterministic (e.g. no RNG or reference external information. See *Oracles* later) and produce the same result on all peers.
  - Determinism guaranteed by sandbox.
  - Determinism ensures contract code returns the same result across all peers.
- Contract code can only access data in the supplied transaction, nothing else.
- Deployed on all peers that are party to the agreement.
- A transaction may have multiple states that refer to multiple contracts (e.g. transaction for swapping *Bond* for Cash).

CONTRACT CODE

---

LEGAL PROSE

STATE

CONTRACT REF	CONTRACT PROPERTIES
PARTICIPANTS	From: Alice
Alice	To: Bob
Bob	Amount: £10
	Due: 31/03/2017
	Fid: ES
	Penalty: 20%

74kguf7974lkjpuj08b... Each state is paired with a contract

Contracts can also refer to a legal prose document, to explicitly state the rules governing the evolution of the ledger in case of legal disputes.

Corda has made the notion of "Code is Not Law" explicit, by allowing a contract to refer to a legal prose that serves as a reference to the real-world legal agreement between parties should disputes arise.

11

Now, coming to contracts in Corda. So, contracts as I mentioned can be written in any JVM language. So, Java, Kotlin there are many others and it is executed in the sandbox which is a modified version of the java virtual machine and like I mentioned before there is contract code and associated with that contract code you can also associate a legal agreement with it and the contract code is going to refer to some state information and some participles, right.

So, each state the states think of it as a record that is going to be paired with a contract. So, a state cannot be manipulated by multiple contracts. Contract code has to be deterministic that is needed for any kind of a distributed ledger, but it can reference external information through oracles. So, we will talk about that in the next lecture.

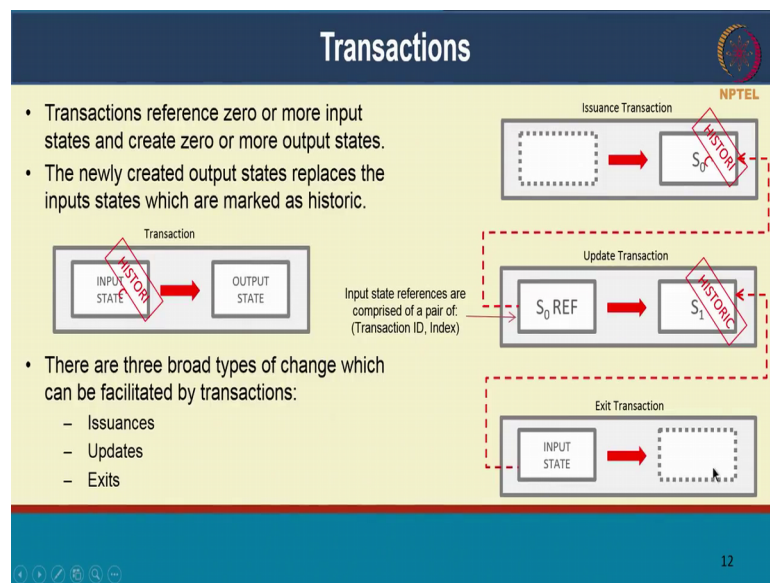
So, the sandbox because it is it is a restricted environment it guarantees determinism. So, that is why that is the modified version of JVM. So, it does not permit any non determinism or randomness to get in. The determinism also ensures that the contract code determines the same result across all the peers. So, each contract is going to be at transaction is going to be executed independently by multiple peers, but the determinism is ensured because the code will always return the same result.

So, the contract is only deployed on the peers that are party to an agreement. So, the other peers will not even know the existence of this contract. The transaction can have multiple states. So, I mentioned the code refers to multiple states. So, it can have

multiple states that refer to multiple contracts, right. So, one transaction can actually operate on multiple contracts and each contract can have multiple states. For instance you could like here is an example right you can have a transaction for swap for swapping a bond for cash you are selling a bond. You are getting cash in return. So, these could be different states that are manipulated by the transaction.

So, I mentioned about this code is not law. So, I would not repeat it again here.

(Refer Slide Time: 22:45)

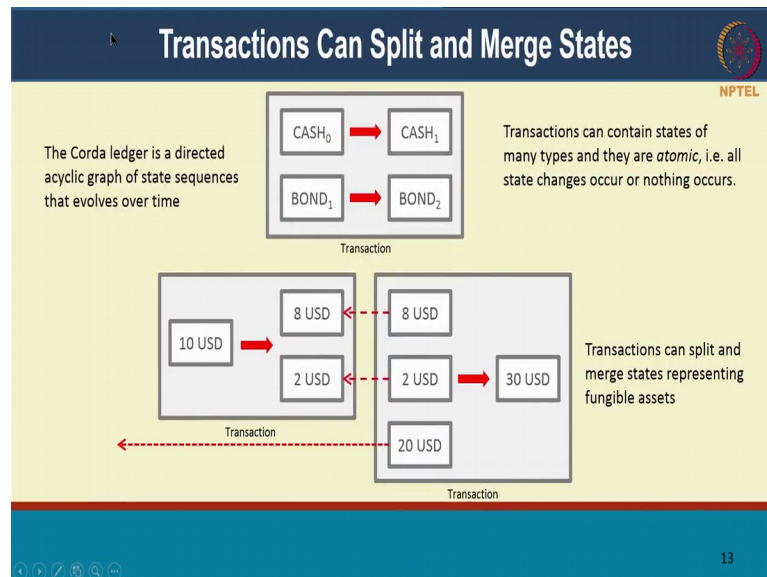


So, coming to what transactions look like. So, transactions reference zero or more output states and creates you sorry reference zero or more input states and create zero or more output states. This is very similar to bitcoin except that state can have a arbitrary information. Newly created output states mark the input states as historic. There are three broad types of exchanges or transactions that are that are specified by Corda. So, these are issuances updates and exists.

So, let us just work through an example transaction flow. So, it could be an issuance transaction where a new state is created from nothing, right. So, you can say only these entities can create an issuance transactions as fine grained access control. So, only maybe a bank can issue let us say a financial instrument. So, then that issuance is created and it is recorded on the blockchain a new state is created and then the next one may be an update. So, the state can go from S 0 to S 1 and then that will mark the previous S 0 as historic. So, that is again just basic use UTXO.

And finally, you could have an exit transaction, where an input state is basically expired or matured basically it is a financial instrument that is now no longer valid. So, you can exit. So, that is also possible.

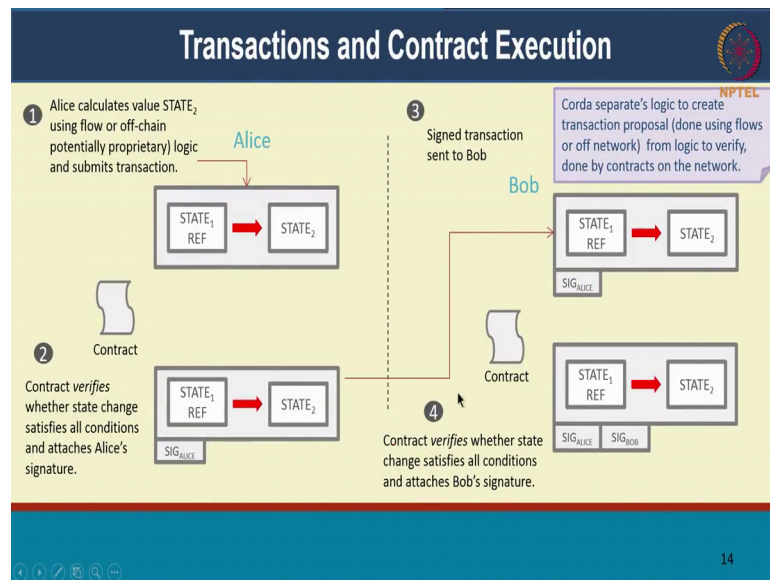
(Refer Slide Time: 24:05)



Now, transactions can split and merge states. So, note that Corda does not have an inherent crypto currency, but you can model cash or bonds or other financial instruments as states. So, the cash itself can be modeled as a state and it can map from let us say one cash state to another cash state. So, cash it can go from cash 0 to cash 1 in exchange for let us say a bond which goes from 1 one to bond 2. So, these are captured.

So, this is again a very similar UTXO example. You have split 10 USD into 8 USD in 2 USD and you have going to consume those in one or more transactions. And, this 20 USD can be pointing to a previous transaction altogether. So, this is similar a split and merge representing fungible assets very simpler to how bitcoin is a fungible asset.

(Refer Slide Time: 24:59)



So, let us just work through an example of our transaction and contract execution will work. So, let us say this is a transaction between two parties Alice and Bob, right. Alice first execute the transaction it calculates the state value. So, it is calculate state 2 using a flow. So, we will talk about transaction flows in just a little bit um. So, it going is going to execute this logic and it is going to submit a transaction. So, that is the first step.

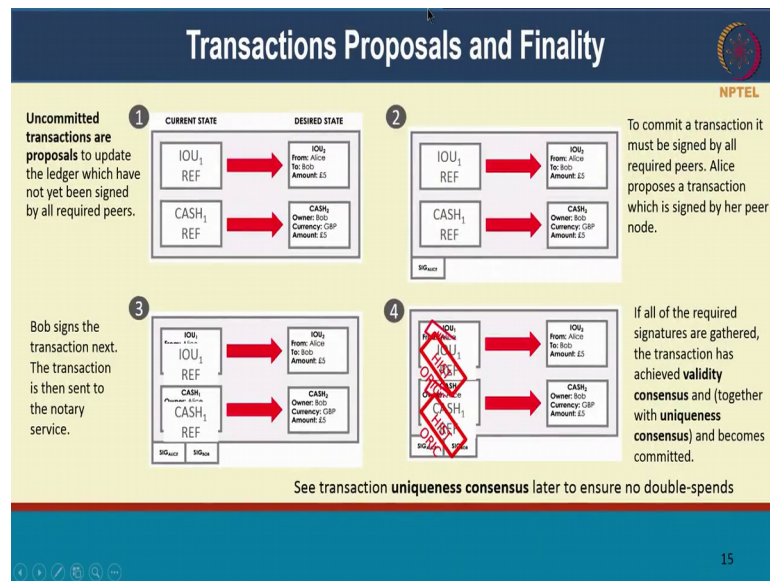
So, second step once the contract gets this transaction from Alice it is going to first verify whether this is a valid transaction and the state change you will check whether all conditions or preconditions for this transaction are satisfied and these preconditions are may are specified in what is called a at a flow. So, this is a we so, we will come to the details of the flow itself, but think that there is a contract and there are for each step of the contract there are preconditions and we the preconditions will specify who needs to sign perhaps, what others information needs to be valid maybe it refers to an oracle, an external piece of information all of that is validated and only if all the preconditions are met will the contract be will the contract execution be valid, right. So, that now includes Alice's signature and that signature is verified. So, that is the contract part.

The third thing is now Bob might receive this as a peer to peer message. So, it receives this message from Alice and then it will check again it can then execute other validation logic to see whether Bob is with signing this transaction and this logic can be some

external information. So, Bob can use information private to just Bob itself and determine whether it wants to sign this and let us say it agrees to sign it then Bob will also sign this thing and the contract can verify that all again all the conditions are met and this is actually a valid transaction.

Now, once this fully goes through then the transaction is recorded by both Alice and Bob that now this is find out.

(Refer Slide Time: 27:11)

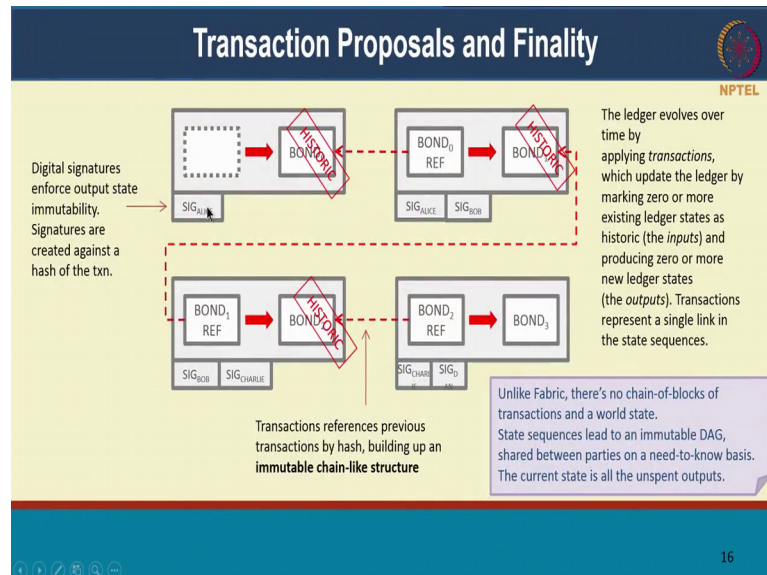


Now, coming to the notion of finality; first there are proposals that are made. So, there is a current state and a desired state. So, Alice creates this desired state submits it to sorry it is goes this way this time. So, submits it to the contract, but to commit the transaction it must be signed by all the peers all the required peers. So, Alice proposes and it is it is signed by her peer not. So, there is signature of Alice that is added to this desired state and then it is sent to Bob and then bob can then figure out whether they want to Bob also sign it and then add Bob's signature.

And, then finally, once the contract verification validation passes it is also determined that this state is unique this the input states are not consumed in other concurrent transactions once it is determined to be unique that is through we will talk about how that uniqueness is achieved uniqueness consensus and then that is recorded as final. So, that is once it is recorded and these states are marked historic is when transaction finality is reached.

So, we still need to see what is this validation there is happening, how is uniqueness achieved right we will go through some of those notions.

(Refer Slide Time: 28:28)

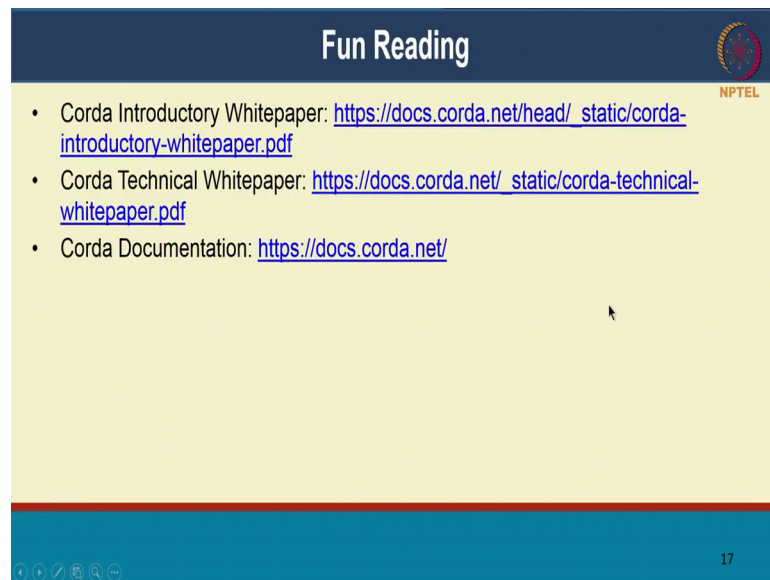


So, the digital signatures are the ones that are giving you immutability and authenticity. So, you know that. In fact, Alice was the one that signed it. No one has modified it since the time Alice signed it. So, when an auditor comes and looks at this they know, they know exactly what has happened they know that none of these could have been altered because if they were altered the signatures would not match, right.

So, through these digital signatures you can get the mutability and authenticity. But, these chain of blocks are only this chain of sorry state information is only available with Alice and Bob it is not available universally with other nodes, that is the no blockchain aspect, right. So, the state sequences lead to almost like a DAG, right. Each node is going to maintain this DAG of state sequences each state could be referring to past state and it is really a DAG, ok.



(Refer Slide Time: 29:21)



**Fun Reading**

- Corda Introductory Whitepaper: [https://docs.corda.net/head/\\_static/corda-introductory-whitepaper.pdf](https://docs.corda.net/head/_static/corda-introductory-whitepaper.pdf)
- Corda Technical Whitepaper: [https://docs.corda.net/\\_static/corda-technical-whitepaper.pdf](https://docs.corda.net/_static/corda-technical-whitepaper.pdf)
- Corda Documentation: <https://docs.corda.net/>

NPTEL

17

So, we will pass at this stage for the first lecture on Corda. So, in the next lecture we are going to look at how the consensus works, how notaries work, oracles as a first class citizen in Corda so, some of the other aspects. So, we have just looked at what a transaction looks like, what the ledger is the fact that ledgers are distinct states stored by peers and with some overlap with the state shared by two peers will be on both ledgers, but there could be state that is that is in one peer and not in the other peer, right. So, it is really a distributed ledger it is not really a blockchain.

So, here in the in our fund reading section there is links to the Corda interactive white paper just a that is just a very high level document maybe 10 pages or so, I think. The technical white paper is much long that is about 60 pages, but it is good information it is it tells you it gives you good motivating set of it is motivating article and why they chose to design it this way, why they feel that blockchains as such are not very good for privacy and you really need this kind of a distributed ledger or where transactions are sent to parties on a need to know basis and others will not have even any semblance of these transactions having happened, ok.

So, with that let us move to the next part of Corda. I will see you soon at the next lecture.