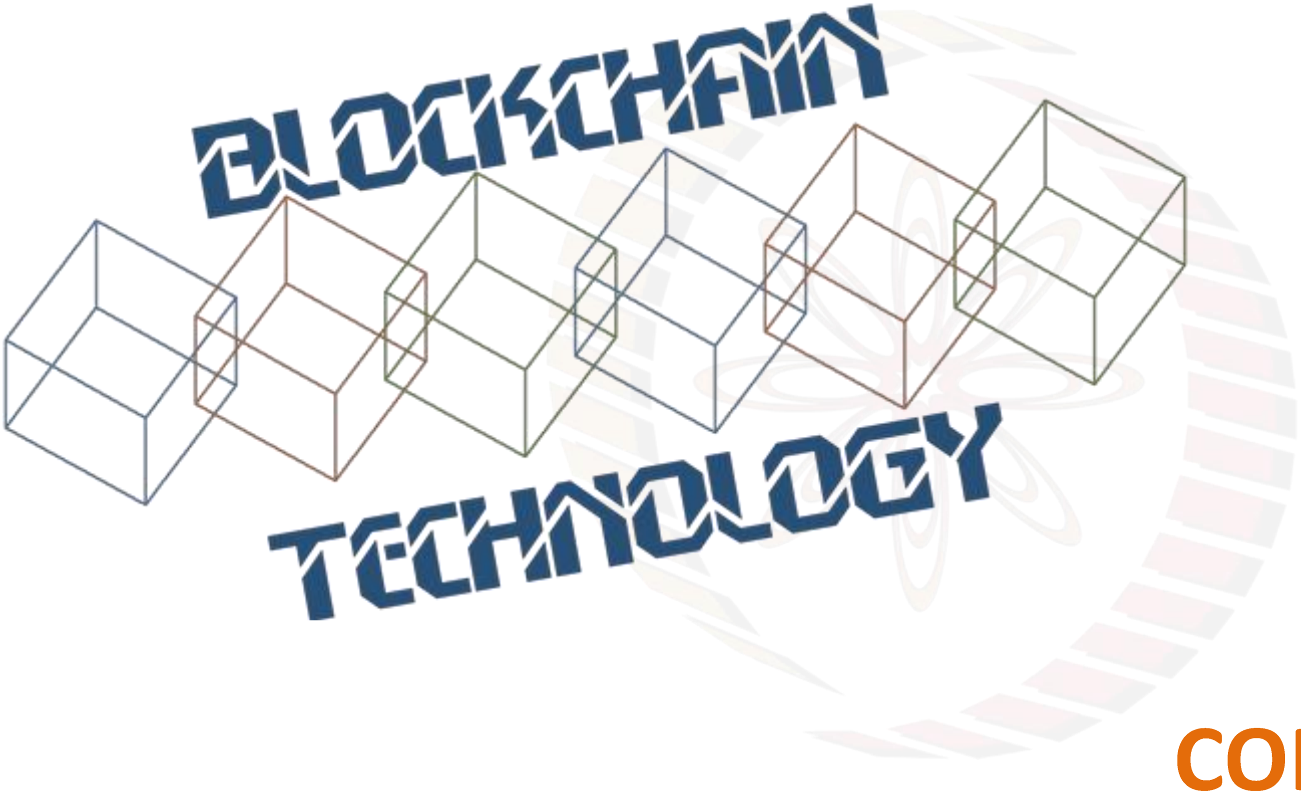
BLOCKCHAINS

ARCHITECTURE, DESIGN AND USE CASES

SANDIP CHAKRABORTY
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

PRAVEEN JAYACHANDRAN
IBM RESEARCH,
INDIA

*Image courtesy: http://beetfusion.com/



CORDA – PART 1

Corda - Brief Overview

- 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.

Legal Agreements as a Foundational Concept

Elements of a Legal Agreement (State)

Issuer: Australia and New Zealand Banking Group

Beneficiary: Scentre Group

Applicant: Smith Co.

Amount: $...

Interest: ..%

Effective date:
- From: May 10th 2018
- To: May 10th 2020

Amount payable to be computed based on share price as of ….

Parties (Identities, Private Ledger)

Time (Time Windows)
- Before (Expiry)
- After (Maturity date)
- Within period

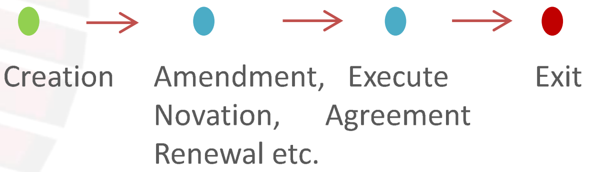
External facts and conditions that govern the contract (Oracles) Corporate Action, Interest rates, share price, bankruptcy, FX conversion etc.

Witness (Notaries, Observers)

Terms and Conditions (Legal Prose)

Supporting documentation (Attachments)

Lifecycle of an Agreement



● → ● → ● → ●

Creation

Amendment, Novation, Renewal etc.

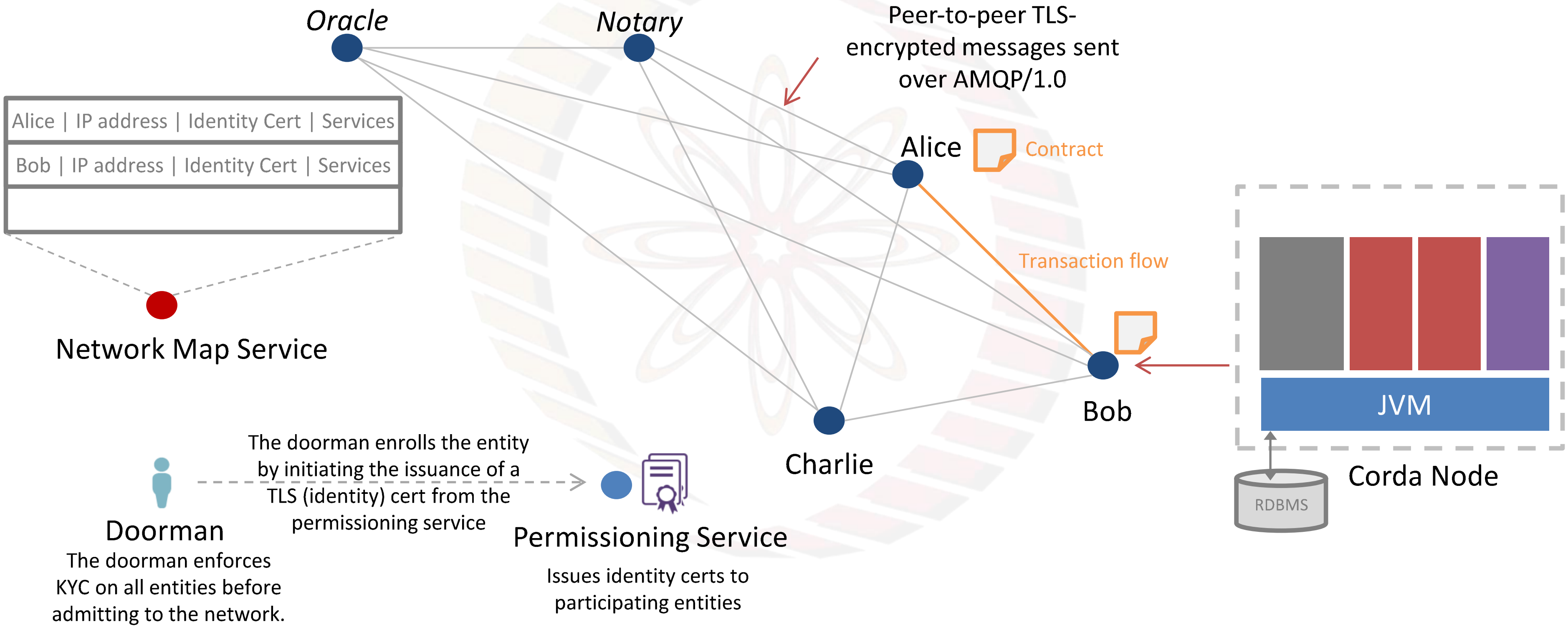
Execute Agreement

Exit

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.

The Corda Peer-to-Peer Network



Oracle

Notary

Peer-to-peer TLS-encrypted messages sent over AMQP/1.0

Alice | IP address | Identity Cert | Services

Bob | IP address | Identity Cert | Services

Network Map Service

Alice

Contract

Transaction flow

Bob

Charlie

JVM

RDBMS

Corda Node

The doorman enrolls the entity by initiating the issuance of a TLS (identity) cert from the permissioning service

Doorman

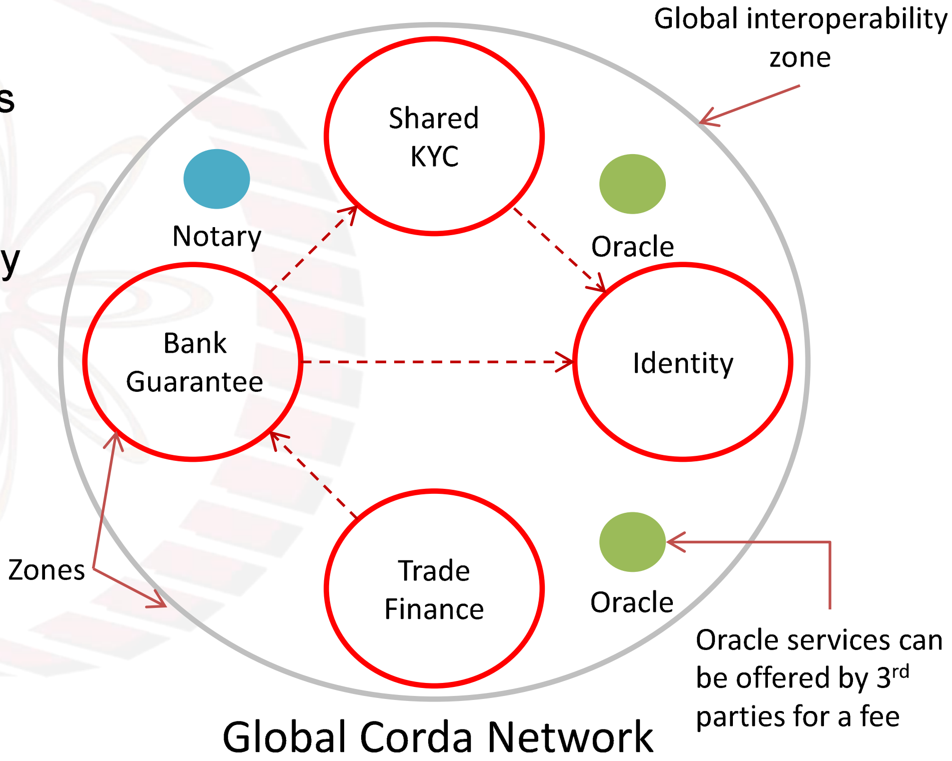
The doorman enforces KYC on all entities before admitting to the network.

Permissioning Service

Issues identity certs to participating entities

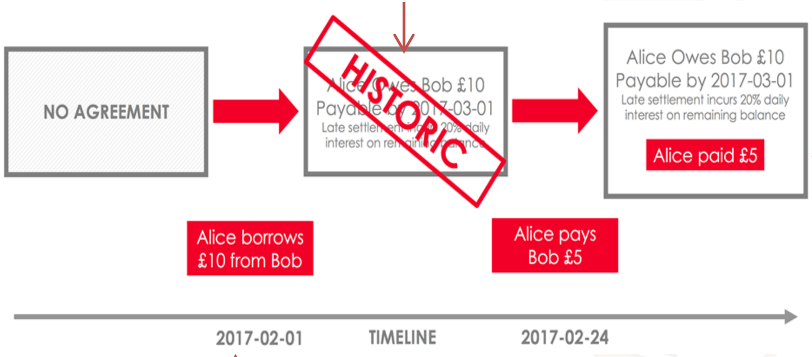
Global Corda Network for Interoperability

- R3 is creating a global shared network for interoperability of permissioned Corda networks
  - Privacy of permissioned networks, with interoperability of public networks.
- A Corda network exists within an interoperability zone and can transact with other nodes in a zone.
  - The global interoperability zone allows nodes to transact across different zones
  - Global Corda Network allows assets to flow across different zones.
- Corda 3.0 delivered stable wire protocol
  - Future releases of Corda will be backward compatible at the network layer.



UTXO State Machine Model

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



NO AGREEMENT

HISTORIC

Alice Owes Bob £10
Payable by 2017-03-01
Late settlement incurs 20% daily interest on remaining balance

Alice Owes Bob £10
Payable by 2017-03-01
Late settlement incurs 20% daily interest on remaining balance

Alice paid £5

Alice borrows £10 from Bob

Alice pays Bob £5

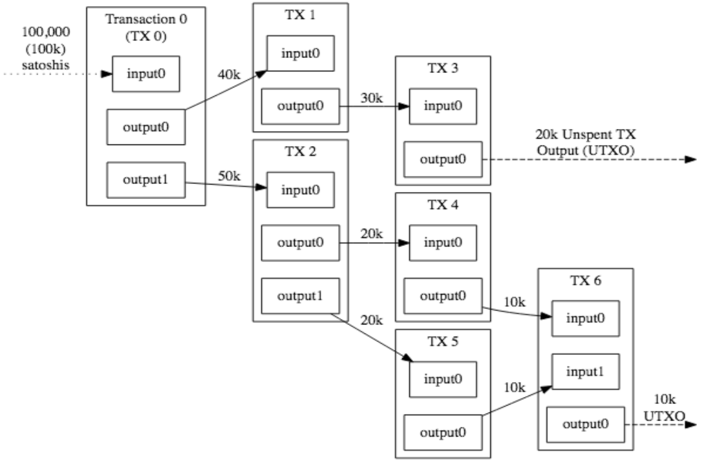
2017-02-01 TIMELINE 2017-02-24

Transaction

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

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

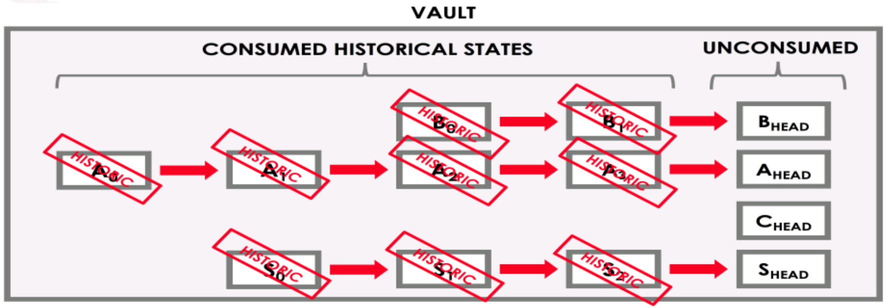
- States are immutable and represent a shared fact at specific point in time.
- States evolve by allowing new states to replace old states, resulting in a state sequence.



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

Vaults and States

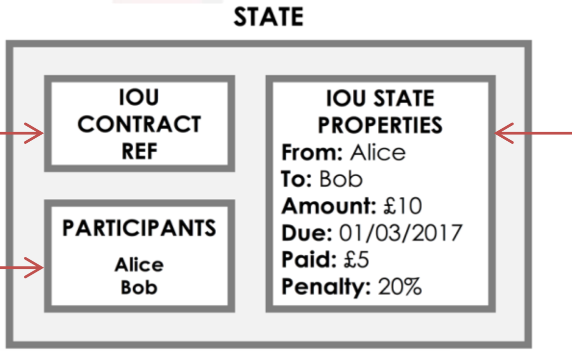
- Each node on the network maintains a *vault* - a database where it tracks all the current and historic states that it is aware of.
- Vaults are currently based on the H2 embedded SQL engine. Other databases supporting JDBC are planned.
- The *current state* of the ledger comprises of all unconsumed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

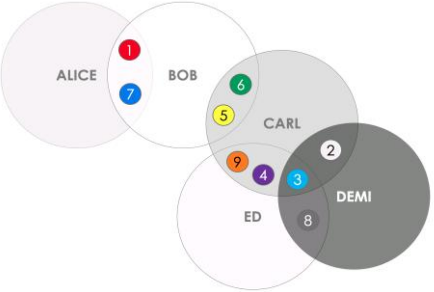
Participants list the peers who can consume this state in a transaction

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.

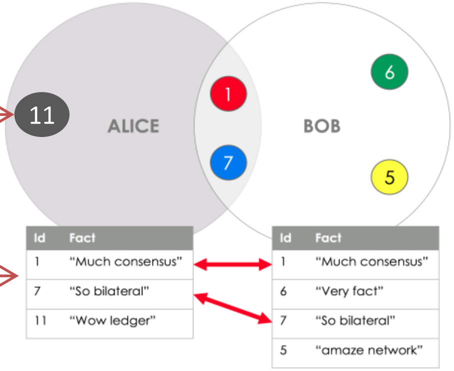
The Corda Ledger

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)



Id	Fact
1	“Much consensus”
7	“So bilateral”
11	“Wow ledger”

Id	Fact
1	“Much consensus”
6	“Very fact”
7	“So bilateral”
5	“amaze network”

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

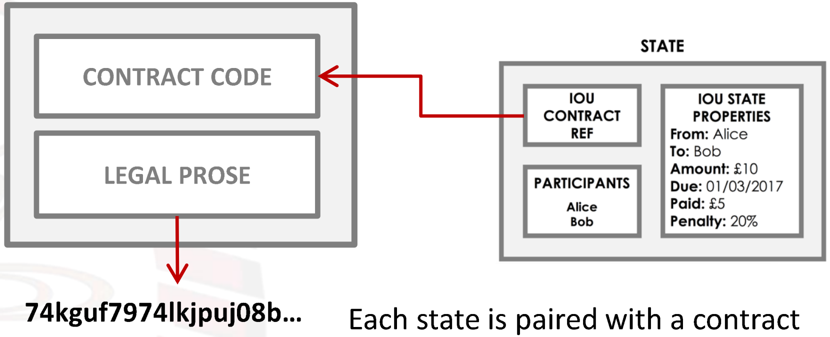
- **No accounts**: Allows for transactions to be applied in parallel.
- **Transaction Ordering**: Transactions ordering is enforced by hash functions that identity previous states.
- **Consensus**: Conflict is a double spend problem.
- **Auditability**: Full history of all activity is recorded.

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).

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*).



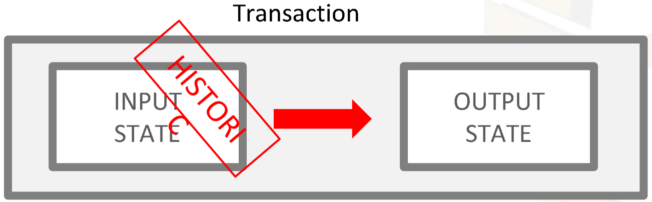
Each state is paired with a contract

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.

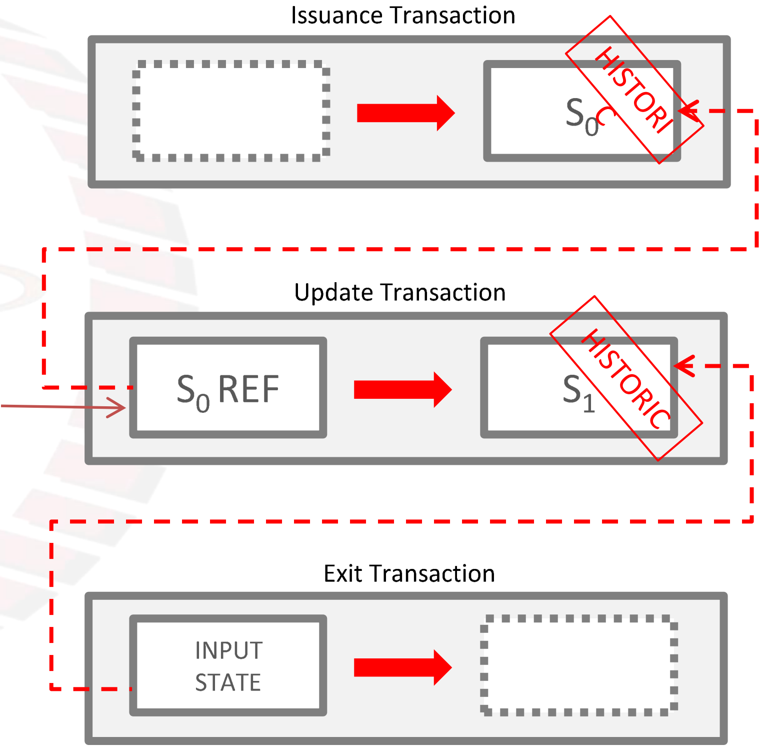
> Corda has made the notion of “Code is Not Law” explicit, by allowing a contract to refer to a legal prose that serves as a reference to the real-world legal agreement between parties should disputes arise.

Transactions

- Transactions reference zero or more input states and create zero or more output states.
- The newly created output states replaces the inputs states which are marked as historic.



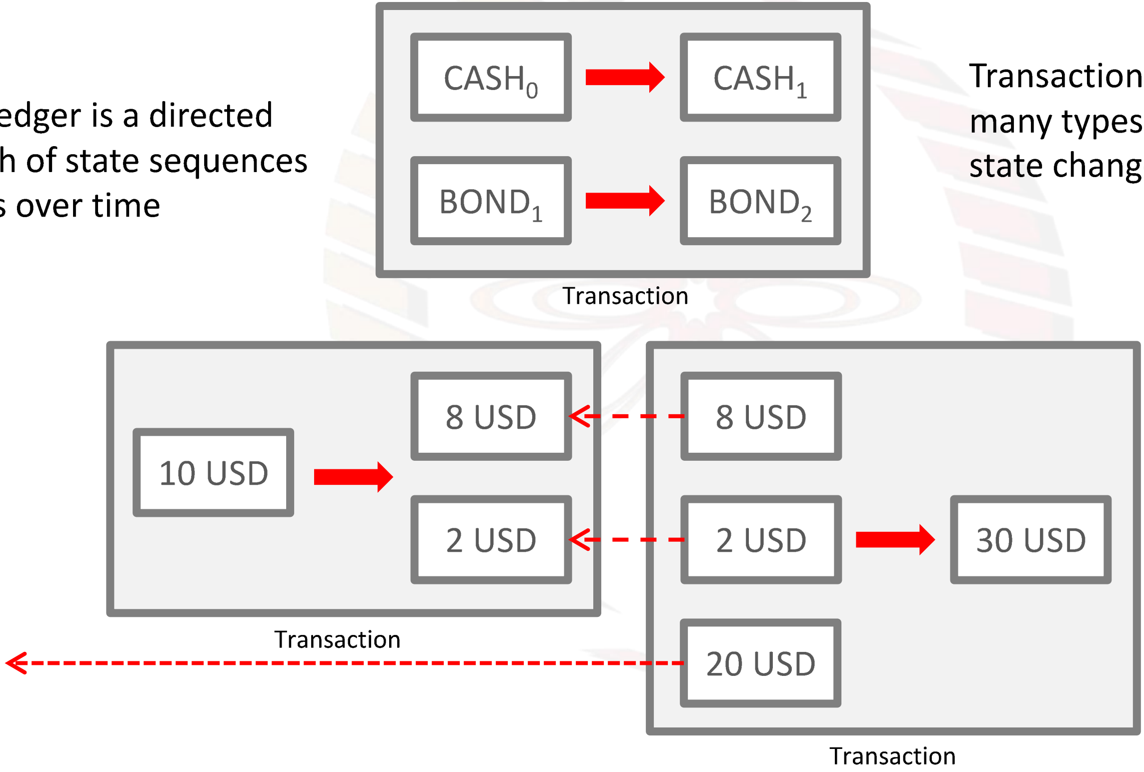
Input state references are comprised of a pair of: (Transaction ID, Index)



- There are three broad types of change which can be facilitated by transactions:
  - Issuances
  - Updates
  - Exits

Transactions Can Split and Merge States

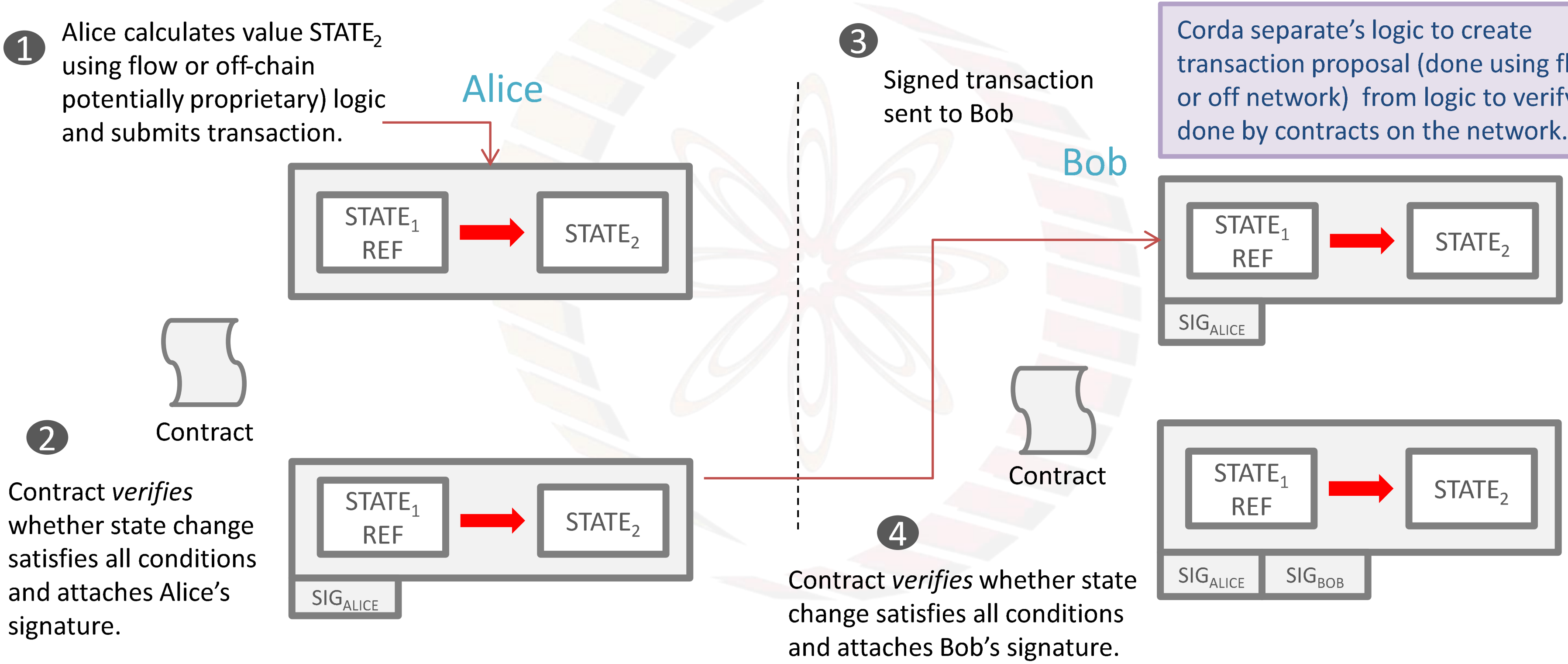
The Corda ledger is a directed acyclic graph of state sequences that evolves over time



Transactions can contain states of many types and they are *atomic*, i.e. all state changes occur or nothing occurs.

Transactions can split and merge states representing fungible assets

Transactions and Contract Execution



1 Alice calculates value STATE$_2$ using flow or off-chain potentially proprietary) logic and submits transaction.

2 Contract *verifies* whether state change satisfies all conditions and attaches Alice's signature.

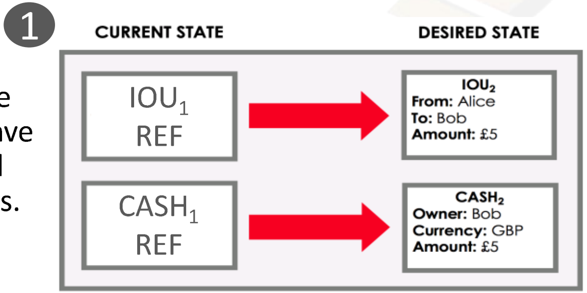
3 Signed transaction sent to Bob

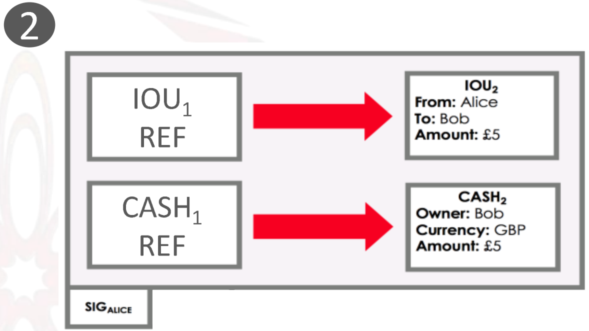
4 Contract *verifies* whether state change satisfies all conditions and attaches Bob's signature.

Corda separate's logic to create transaction proposal (done using flows or off network) from logic to verify, done by contracts on the network.

Transactions Proposals and Finality

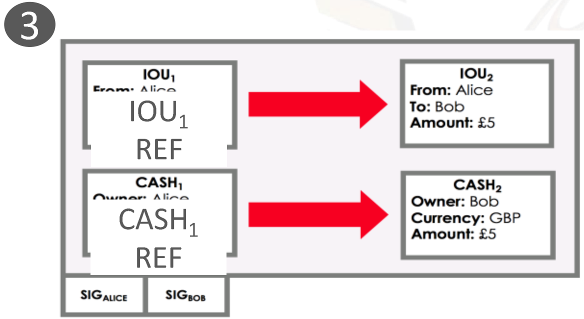
Uncommitted transactions are proposals to update the ledger which have not yet been signed by all required peers.

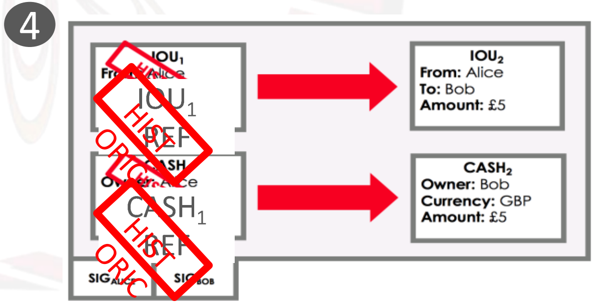




To commit a transaction it must be signed by all required peers. Alice proposes a transaction which is signed by her peer node.

Bob signs the transaction next. The transaction is then sent to the notary service.



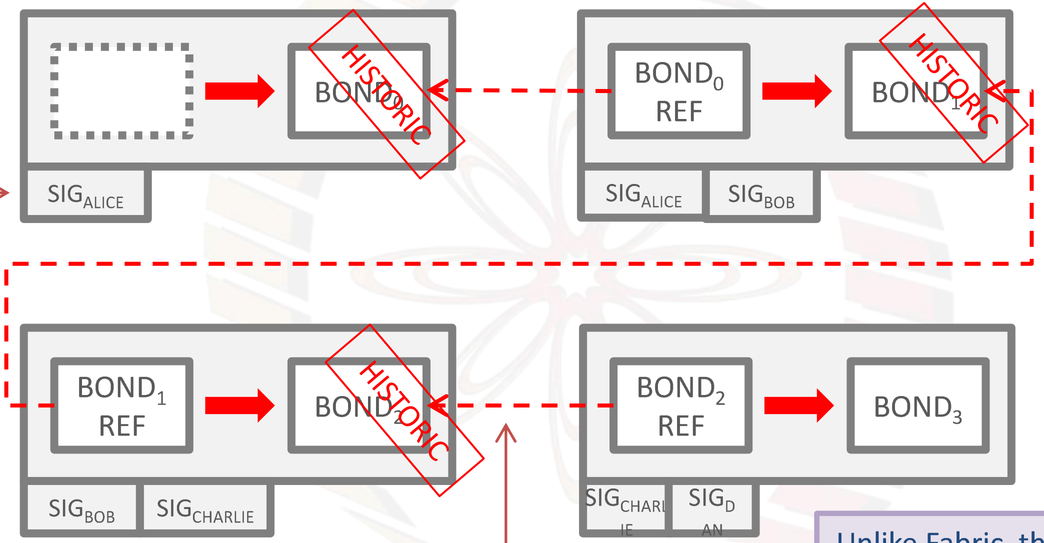


If all of the required signatures are gathered, the transaction has achieved **validity consensus** and (together with **uniqueness consensus**) and becomes committed.

See transaction **uniqueness consensus** later to ensure no double-spends

Transaction Proposals and Finality

Digital signatures enforce output state immutability. Signatures are created against a hash of the txn.



The ledger evolves over time by applying *transactions*, which update the ledger by marking zero or more existing ledger states as historic (the *inputs*) and producing zero or more new ledger states (the *outputs*). Transactions represent a single link in the state sequences.

Transactions references previous transactions by hash, building up an **immutable chain-like structure**

Unlike Fabric, there's no chain-of-blocks of transactions and a world state.
State sequences lead to an immutable DAG, shared between parties on a need-to-know basis.
The current state is all the unspent outputs.

Fun Reading

- Corda Introductory Whitepaper: https://docs.corda.net/head/_static/corda-introductory-whitepaper.pdf
- Corda Technical Whitepaper: https://docs.corda.net/_static/corda-technical-whitepaper.pdf
- Corda Documentation: https://docs.corda.net/

thank you!