8/31/Teleconference

Attendees:

- Ram Jagadeesan
- Murali
- Hart
- David
- Stan
- Alex
- Sri

Agenda:

- Security Functions Continued
- Bootstrap
- Identity & Policy Modules

Identity Requirements: Are we bootstrapping identities from scratch on the blockchain, or is there a requirement for using an external/legacy identity service?
David's original membership services model allows user identity to be imported from an external authority.

Federated Identity: Example DTCC has a node - DTCC identity service is responsible for authentication, authorization of DTCC entities, similarly each node/domain has it's own service.

System entities are known/public- will not be anonymous. In the future do we need to consider pseudo-anonymous system entities?
We can rely on PKI backbone of the system - have a root-CA within the system - Can designate an entity to authorize additions.

"Selective Release"

8/17/2016

Attendees:

- Ram Jagadeesan
- Vipin Bharathan
- Stan Liberman
- Binh Nguyen
- Murali
- Mic Bowman
- David Kravitz

Agenda:

- Agenda Review
- Security Requirements
- Bootstrap process
- Identity Services
-

Notes:

- Future agenda topics Storage Abstraction - Binh and Vipin to have offline discussion
- # <EnrollmentID>: <system_role (1:client, 2: peer, 4: validator, 8: auditor)> <EnrollmentPWD> <Affiliation> <Affiliation_Role> <JSON_Metadata>
- Fabric membership services config file https://gerrit.hyperledger.org/r/gitweb?p=fabric.git;a=blob;f=membersrvc/membersrvc.yaml;h=f4567bc9c5a66323e710a0d4bf55c296369bea44;hb=refs/heads/master
-

Security Requirements:

- No single-point of failure
- No single point of trust

Identity Services:

- Authenticate system entities - identity and credentials
- Registration - Enrollment - vetting
- Authorization and entitlement (permissions)

Policy Services:

- Group policies for chain
- Identity, enrollment, entitlement policies
- Policy management
- Interfaces to policy enforcement points

Bootstrap Process:

- Fabric -
    - 1) Static configuration file - specified in config file. Identities and properties are specified in the config. “Register” the id’s in the system. The Peer would use the identity to enroll with Membership services. Peer gets Ecert (Enrollment cert) from MS, and uses the cert for authentication with other entities. Peers can then obtain and use Tcerts (Transaction certs) for tx for privacy.
        - Config structs:
            - Id
            - System Role (1, …,8)
            - OTP
            -
    - 2) API - Admin application - registration API and enrollment API. An authorized entity would use the the API to register new entities.
    - Registrar entity needs to be configured. After the registrar is up, it can support dynamic enrollment.
    -

- Sawtooth:
    - Use Ledger for managing services. Specific transaction family for registration and policies.
    - Rights and privileges based on validation information
    - Genesis validator - retired after bootstrap - accepts persistent validator.
    - Follow-on registration - validation using PoET.
    - Special privileges constrained in time; does not extend beyond bootstrap.

- Common: Registration, enrollment, entitlement
- Pros and Cons of distributed launch vs simpler single trusted bootstrap service?

- Mic’s proposal: The ledger itself could serve as the registry
- AIs: Mic and Binh to post documentation. Mic is on vacation for two weeks starting next. May be late for text. Team to review and continue discussion over e-mail and next meeting.
-

8/11/16 Documentation Working Session - Do we need notes, or shall we work on the doc?

Agenda: Work on Arch Doc:
https://docs.google.com/document/d/1bdr4MdnpUvv_3jN5DYSNp1ctjKGPH7K_yiMSQYcwBBo/edit#

Attendees:

- Ram Jagadeesan
- Mic Bowman
- Stan Liberman
- Hart Montgomery
- Binh Nguyen

Notes - captured on working doc.

Binh and Mic volunteered to flesh out the Consensus section, and Ram, Hart and Stan volunteered to flesh out the Smart Contract section.

8/3/16 Teleconference

Attendees:

- Ram Jagadeesan
- Mic Bowman
- Hart Montgomery
- Stanislav Liberman
- Alexander Yakovlev
- Craig Rowe
- Vipin Bharathan
- David Kravitz
- Jonathan Levi

Agenda:

Documentation
Security & Privacy

Documentation Plan:

Start with Google Doc for Architecture, and see at what point it's mature enough for formal doc using Latex

Start a Google Doc folder for multiple docs

Stan: Suggest a high-level architecture module definition for the documentation
Ken: Prioritize top-10 security requirement
Mic: Expediency of implementation vs architectural alignment

AI:

Ram will start Arch Google Doc, and an Arch-WG-doc folder

F2F 7/27/16 General Session

Attendees:

- Ram Jagadeesan
- Mic Bowman
- Binh Nguyen
- Tamas Blummer
- Hart Montgomery
- Stanislav Liberman
- David Kravitz
- Jonathan Levi
- Christopher Allen
- Chris Price

Regarding the information that persists…
We need to keep as part of the permanent record the endorser request and response, and the policy that was used to approve based on the results. This information may be useful later for audit or endorser policy decisions.

Does the transactor get to choose the endorser set or does a more general policy (e.g. from the chaincode) determine the endorser set? If the transactor chooses the endorser set then there must be a mechanism for dynamically creating the “privacy” capabilities. Per Hart: both unlinkability and selection of endorsers might be requirements. Architecture must accommodate both requirements.

DK - Audit is done through key-delegation. Doesn’t need to be explicit in “endorsement policy”

Security & Privacy Requirements:

Genesis - Bootstrap Requirements and Trust Model:
Need input on this.

Simplest starting-point: Is single entity - single trusted party (CCP) who is sole admin and validator, and writes transactions to the BC. Genesis ID - Root of trust - Bootstrap identity and policy system. Can delegate authority.
One common approach is to record public-key in the genesis-block of the BC, and subsequent (validator/endorser) identity registrations/delegations could also be recorded in the BC.

Requirement for trusted setup - without trusting a single part? E.g. Bootstrapping a consortium BC.

Should the ledger be used to record.

Scope: Core Identity/Authorization for validators/endorsers and BC system entities. We need to define enrollment of system entities.
Could we leave the identity/authorization/policies for the transactors/participants to the smart-contract layer?

Req: Dynamic registration of new system entities
Req: Policy for system entities adds/drops
Req: Audit policy and explicit system audit roles/nodes registration that is visible? Auditor - visibility into transaction details (whether or not transactors wants that).
R

7/26/16 F2F Meeting Documentation Session - 1 PM PT

Agenda: Outline of the documentation for consensus layer, smart-contract layer and interface specifications

Attendees:

- Ram Jagadeesan
- Hart Montgomery
- Stanislav Liberman
- Mic Bowman
- Binh Nguyen
- Chris Price
- Nikilesh Subramoniapillai Ajeetha

Recap of Consensus & Business Logic:

Smart-contract layer: Validates and executes transactions, and computes state-deltas, according to validation policy, and maintains the global-state.

Consensus layer: Confirms the correctness of all transactions in a proposed block, according to validation and consensus policies. Agreement is on order and correctness and hence on results of execution (implies agreement on global state). Interfaces and depends on smart-contract layer to verify correctness of an ordered set of transactions in a block.

Process:

1. Select initial state
2. Viability Test - set of rules for correctness given current state (prior)
3. Compute State Delta and set of events
4. Order
5. Verify Independence of State Updates (Viability/Correctness of the Block?)
6. Commit transactions and fire events

Functional Description of Smart Contract Layer

- Determines the validity of a transaction given a starting state. It must reject the transactions found to be invalid.
- Part of the validity test includes the permission test (e.g. authorized by asset owner)
- The input state includes the data state for the smart contract (transaction family) and information about the state of the chain (such as block number),

- For valid transaction, computes the delta on the state as a result of executing the transaction. May generate a list of events that will fire when/if the transaction is committed.
- We assume that transactions are atomic. If we have dependent actions that must be committed together, assume they are captured in a single smart contract.
- A node participates in validating a transaction only if the node has permission and access to perform the validation as specified in an “endorsement” policy. The endorsement policy specifies the set of nodes that need to validate a transaction.
- Endorsement policy defines correctness

Interfaces of Smart Contract Layer

- Request to Smart Contract Layer
    - Transaction
    - Transaction dependencies
    - State
- From Smart Contract Layer to Consensus
    - Accept/Reject
    - Read and Write of State
    - State delta
    - Transaction
    - Transaction dependencies (might be richer than input)

      If (committed) State was the only context for transaction validation, then a chain of transaction could not be processed at a faster than one-by-one at block creation rate. Read and Write State will be captured by intercepting requests to State. Similar to transaction dependencies they formulate the assumptions made during validation. Consensus will be able to detect and exclude transactions that were valid under contradicting assumptions.

Functional Description of Consensus Layer

- Confirms the correctness of all transactions in a proposed block, according to endorsement and consensus policies.
- Agreement is on order and correctness and hence on results of execution (implies agreement on global state).
- Interfaces and depends on smart-contract layer to verify correctness of an ordered set of transactions in a block.
- Provides proofs/hash of global-state to allow late joining (or out of sync) nodes to catch-up.
- May support interfacing to eventing function to support events associated with committed transactions and block-level events
- Placeholder for Identity of validators/consenters and associated policy

Interfaces to Consensus Layer

Detailed Process Flow Description
Message Sequence Chart

Next Steps/ToDos:

- Need to talk about state sharing to endorsers and implications of endorsement policy.

7/20/16 Teleconfence

Attendees:

- Ram Jagadeesan
- Mic Bowman
- Hart Montgomery
- Chris Price
- Stanislav Liberman
- Vipin Bharathan

Agenda

- Documentation plan
- Security & Privacy - Functional requirements
- Security Review
- F2F Agenda
- Prioritizing other Agenda topics:
    - Working with W3C, IETF other standards bodies
    - Interworking between ledgers/chains
    - UTXO vs account-state models

Agenda for F2F:

- Documentation - Skeleton - and outline - 3 hours
- Security & Privacy - 3 hours
- Storage layer - DB
- Prioritizing other topics

Documentation Plan:
Hart: Important to capture detailed documentation especially for security review pov.
Mic: Have a working session on documenting arch-wg conclusions

7/6/16 Teleconference

Attendees:

- Ram Jagadeesan
- Christopher Allen
- Hart Montgomery
- Marko Vukolic (mvu@zurich.ibm.com)
- Tamas Blummer
- Stanislav Liberman

Recap of Consensus & Business Logic:

Smart-contract layer: Validates and executes transactions, and computes state-deltas, according to validation policy, and maintains the global-state.

Consensus layer: Confirms the correctness of all transactions in a proposed block, according to validation and consensus policies. Agreement is on order and correctness and hence on results of execution (implies agreement on global state). Interfaces and depends on smart-contract layer to verify correctness of an ordered set of transactions in a block.

Process:

7. Viability Test - set of rules for correctness given current state (prior)
8. Compute State Delta
9. Order
10. Verify Independence of State Updates (Viability/Correctness of the Block?)
11. Commit

Other Agenda Items

- Report out on request to do Consensus APIs from W3C Blockchain Workshop
    - https://twitter.com/ChristopherA/status/748604981259943940
    - Ethan Buchman <ethan@tendermint.com> or <ethan@coinculture.info>
        - http://tendermint.com/blog/tendermint-socket-protocol/
        - https://github.com/tendermint/tmsp
- How to move forward on academic review and security review requirements for cryptography, architecture & APIs, and identity/privacy/confidentiality?
    - How about security review requirements?
        - Functional requirements (a core architecture WG task)
        - Should we providing guidance on our approaches for early academic and security reviews before code audit. Security review framework?

        - Review and Audit (is this an architecture WG task?)
        - What are the stages of security review / levels in the release process?
            - Bitcoin is a running network, Hyperledger is not
    - Fabric team desires to submit papers to academic
        - Can we get details on what papers? Where being submitted? Can we do early review.
    - What are the requirements for a public hyperledger blockchain instance? Brainstorm around this and what it entails?
        - Functional requirements and format details
    -

7/1/16 Teleconference

Attendees:

- Ram Jagadeesan
- Kostas Christidis
- Stanislav Liberman
- Hart Montgomery

Agenda:

- Consensus and Smart-Contract layer discussion continued

Smart-contract layer: Validates and executes transactions, and computes state-deltas, according to validation policy, and maintains the global-state.
Consensus layer: Confirms the correctness of all transactions in a proposed block, according to validation and consensus policies. Agreement is on order and correctness and hence on results of execution (implies agreement on global state). Interfaces and depends on smart-contract layer to verify correctness of an ordered set of transactions in a block.

6/22/16 Teleconference

Attendees:

- Ram Jagadeesan
- Mic Bowman (intel)
- Craig Rowe
- Arnaud Le Hors (IBM)Binh Nguyen
-
- Vipin Bharathan
- Chris Price
- Hart Montgomery
- Sri Krishnamacharya

Viability Test - set of rules for correctness given current state (prior)
Compute State Delta
Order
Verify Independence of State Updates (Viability/Correctness of the Block?)
Commit

Immediate finality vs Probabilistic finality with roll-backs/journalling

6/17 Teleconference

Attendees:

- Ram Jagadeesan, Cisco
- Christian Cachin, IBM
- Chris Price
- Alexander Yakovlev
- Mic Bowman
- Vipin Bharathan
- Hart Montgomery
- Tamas Blummer
- Christopher Allen

Goal:

- Pluggable consensus module/layer - allow various consensus algorithms to be supported using a “unified interface or framework”
- Requirements 1) Agree on order 2) Agree on correctness 3) Support confidentiality
- Try to minimize the dependencies between consensus and business-logic/smart-contract layer and make the dependencies explicit
- Policy driven placement of sub-functions - components.

Chris Allen - Opportunity for new attack surfaces by separating “correctness/endorsement” from consensus?
HM: Decoupling is a security bonus rather than a minus.

Contract -> Meta-data : Deterministic, reversible, side-effects?
Bindings: e.g. business-logic computation is non-reversible (won’t work with probabilistic consensus approaches which require rollbacks)

MIC: Here’s one definition:

Formal requirements for a consensus protocol may include:

- *Agreement*: All correct processes must agree on the same value.
- *Weak validity*: If all correct processes receive the same input value, then they must all output that value.
- *Strong validity*: For each correct process, its output must be the input of some correct process.
- *Termination*: All processes must eventually decide on an output value

Dependencies - correctness - smart-logic/endorsement/validation - depends on state:

Possible means to determine correctness:

- By formal specification of acceptable states (e.g. database consistency constraints)
- By functional computation (i.e. the contract itself determines correctness)
- By external observation (e.g. through the aggregate vote of a set of endorsers)

Showing how Bitcoin’s Nakamoto consensus is equivalent to (distributed-computing) consensus:

- Juan A. Garay, Aggelos Kiayias, Nikos Leonardos:
  The Bitcoin Backbone Protocol: Analysis and Applications. EUROCRYPT (2) 2015:

281-310
http://eprint.iacr.org/2014/765

- Andrew Miller's paper: https://socrates1024.s3.amazonaws.com/consensus.pdf
- Analysis of the Blockchain Protocol in Asynchronous Networks
  Rafael Pass and Lior Seeman and abhi shelat
  https://eprint.iacr.org/2016/454
-

CA: DAO bugs and R3 composability - business logic with deterministic proofs:https://github.com/WebOfTrustInfo/ID2020DesignWorkshop/blob/master/topics-and-advance-readings/DexPredicatesForSmarterSigs.md

For next week's agenda: some questions from identity WG on security and cryptographic review, and how that connects to statements re: release quality (separate from exit criteria). For instance, formal review of the ECDSA derivations used in Membership Services that have never been used before. Is this the role of Architecture or a separate group? So far requirements WG has not been at that level.

6/8 Teleconference

Attendees:

- Ram Jagadeesan
- Craig Rowe
- Jean Safar
- Stan Liberman
- Greg Haskins
- Tamas Blummer
- Arnaud Le Hors
- Mic Bowman
- Chris Price
- Hart Montgomery
- Binh Nguyen
- Marko Vukolic
- Vipin Bharathan

Agenda:

- IBM proposal for new consensus protocol
- Generalization of our basic approach to support “validating set”
- Confidentiality Requirements
- Trust Model - Transitive trust

IBM Proposal Discussion:
“Consensus service” is a common/utility service, that “channels” (private sub-groups - or virtual network) can use. Endorsement policy could specify only two endorsers - for bilateral contract/tx.
Two levels of confidentiality 1)tx-payload only goes to endorsers, other peers do not see the payload - only hash of tx-payload is included in blob to other peers.

Partitioning: particular class of state - state is immutable - create new objects, 2) Semantic level of partitions - sequential, 3)Make dependencies explicit - longer term cryptographic

Granularity of data-model - single object. Leader based.

Committing peer - committing- tx upate can be propagated in cipher-text.
Confidentiality flavors - endorsers see everything in plain-text.

Generalization of basic approach:
Validating-set: Set of validating peer nodes that have cleartext access and authorization to execute and validate transactions
Validation-policy: Rules on what constitutes a valid-tx (similar to endorsement policy)

Smart Contract Function:
Receives tx from communication layer (unicast from client-node)
If this peer is part of validating-set, executes the validation/business-logic and stores result in scratch-space, if tx is valid. Computes state-update, and optionally includes it in tx sent to network. (In this model, it seems we may not need version-dependencies). Tx is rejected/tossed if not-valid.
Transmit tx to network (broadcast to peers) via communication layer
Block-creator (leader) role: Creates proposed block – ordered list of “valid" tx. Uses state-update information to eliminate potentially conflicting txs (e.g. Double spend txs would be updating using same key.). Sends block to Consensus layer.
Block-validator role: Executes each tx in received block that it is authorized to (I.e. This peer belongs to validating set of tx) and appends its “validation(endorsement)” to each valid tx. Checks state-update info for inconsistencies for other txs, that it can not execute. Sends validation responses to Consensus layer.
Block-committer: Commits all state-updates on completion of consensus
Consensus Function:
Block-creator (leader) role: Receives proposed block from SC function and starts consensus protocol to peer-nodes (broadcast) via communication layer. Ensures validation-policy is enforced.
Block-validator role: Receives proposed-block from peer via communication layer, and send to SC function for block-validation – Uses response in Consensus process and confirms that validation-policy is enforced (Additional round of communication may be needed for confirmation.)
Output of Consensus is agreement on ordered set of valid tx. If Block contains hash of global-state, also verifies global state is consistent. If Consensus is not reached tx returned to “mempool”?

5/25 Teleconference

Attendees

- Christopher Allen
- Hart Montgomery
- Mic Bowman
- Ram Jagadeesan
- Greg Haskins
- Tamas Blummer
- Stanislav Liberman
- Marko Vukolic
- Vipin Bharathan
- Vivian Shen

Agenda:
1) Recap of working session discussions
2) Requirement for confidential transaction
3) Continue discussion on Consensus and Smart-Contract modules functional decomposition and interface definition.

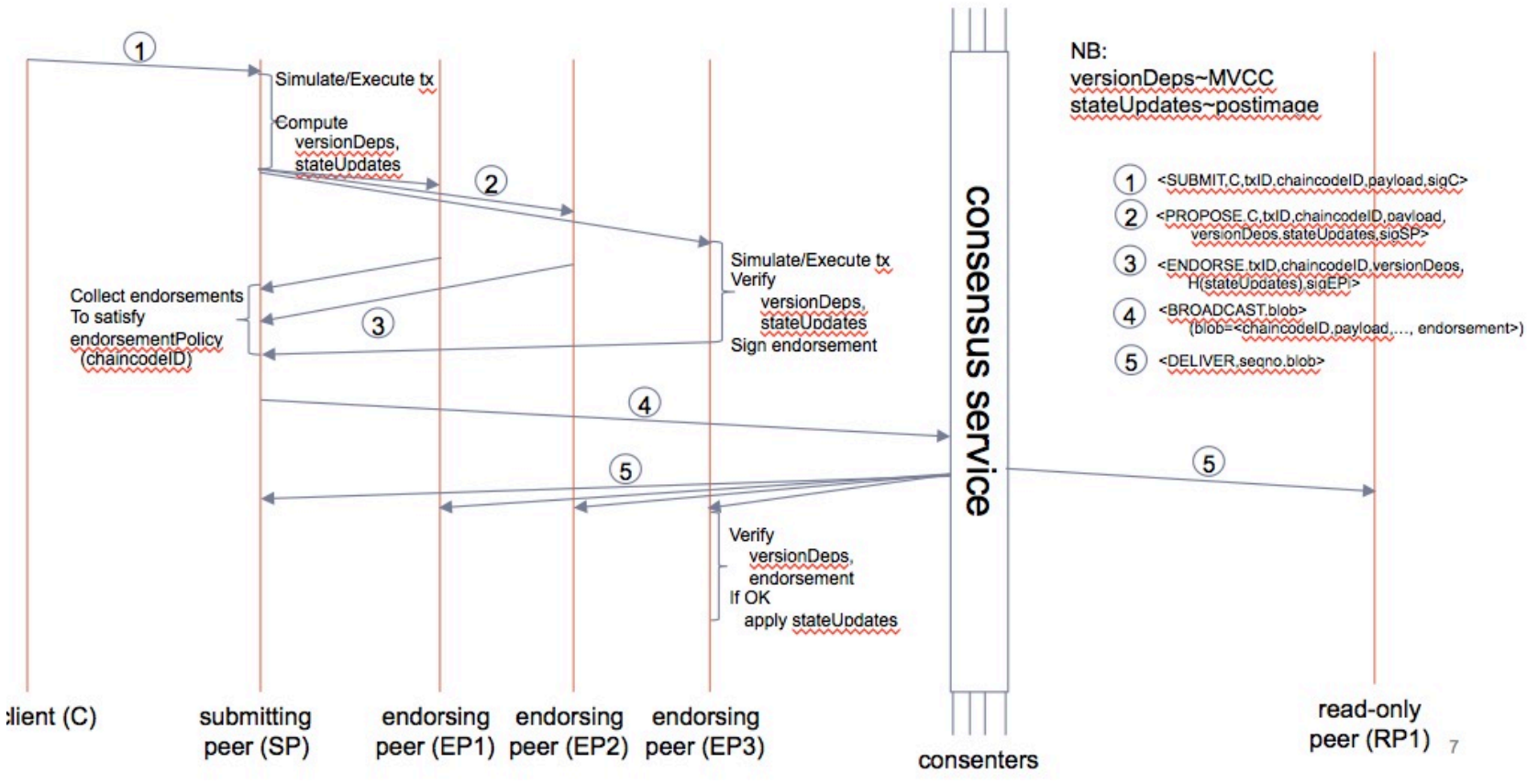
Notes from offline working sessions:
Smart-Contract/Validation and Consensus functions description modeled after bitcoin blockchain and Sawtooth lake:

1. Receives transaction (tx) from communication layer (unicast from client-node)
2. Validates tx – executes the validation/business-logic and stores result in scratch space, if tx is valid. If invalid tx is tossed
3. Transmits tx to network (broadcast to peers) via communication layer
4. Block-creator: creates blocks and validates – by executing tx in-order (check for inconsistencies/double-spends within block, and tosses any invalid txs). Block could contain the hash of global-state.
5. Sends validated block to Consensus module
6. Block-Validator: Receives transaction from Consensus module and executes tx -in–order and validates block, sends response to Consensus.

Consensus module:
1. Block-creator: Receives validated block from SC function and starts consensus protocol to peer-nodes (broadcast) via communication layer.
2. Block-validator: Receives proposed-block from peer via communication layer, and send to SC function for block-validation – Uses response in Consensus process (e.g. Votes Yes if block is valid)

3. Output of Consensus is agreement on ordered set of valid tx. If Block contains hash of global-state, also verifies global state is consistent. If Consensus is not reached tx returned to “mempool”?
What are failure modes and how do we recover?

IBM fabric proposal to handle confidential transactions: 2-phase approach with endorsing peers who endorse (pre-validate) tx that are received from submitting peer. Endorsers have clear-text access and can execute and validate tx. A submitting peer which receives sufficient number of endorsements to meet the endorsement policy will include tx in 2nd phase block-generation at consensus services. The consensus service does not have clear-text access to tx data, and hence cannot validate tx – it only creates an ordered list of tx for inclusion in a block. (Need a second pass validation to eliminate double-spend tx’s within a proposed block). This proposal will be described shortly in fabric docs in the wiki.



Confidentiality Requirement: Only a smaller subset of nodes should have access in clear-text to data that is needed to validate a tx. (forward secrecy/confidentiality)

Mic: 2-phase may have a performance impact, when we try to meet confidentiality requirements
Marko: Parallelization could improve performance.

What happens when chain-code's interact with other chain-codes with different endorsement policies? Generally problems/complexity with crossing confidentiality domains.

(BTW, a link to Blockstream Elements Confidential Transactions zero-knowledge method. It does have a computational cost such that it can't be use for all use cases. https://elementsproject.org/elements/confidential-transactions/)

CA: I am bothered a bit to know what use cases need what levels of confidentiality, and what the transaction speed requirements are needed for those cases.

The idea of separating correctness-testing/validation from consensus functions is desirable, where each node is not doing everything.

Should it be data-centric or code-centric?

Any requirement for confidentiality for code?

Can endorsement be done on an enclave?

F2F Meeting 5/6/16

Attendees

- Christopher Allen, Principal Architect Blockstream <ChristopherA@Blockstream.com>
- Ram Jagadeesan, Cisco
- Jan Willem Barnhoorn, ABN AMRO
- Tim Blankers, ABN AMRO
- Vipin Bharathan, BNP Paribas Americas
- Stanislav Liberman, CME Group
- Takahiro Inaba, NTT
- Keith Smith, IBM
- Mark Moir, Oracle Labs
- Binh Nguyen, IBM
- Vani Komera, DTCC
- Murali, DTCC
- Mike Haley, AlphaPoint
- Mic Bowman, Intel
- Cian Montgomery, Intel
- James Mitchell, Intel
- Shawn Amundson, Intel
- Simon Schubert, IBM Zurich (call-in)

- Tamas Blummer, Digital Asset
- Renat Khasanshyn, Altoros
- Jatinder Bali, Citigroup
- Hart Montgomery, FLA (call-in)
- Sumabala Nair, IBM

Agenda

- Intros
- Consensus layer/module - interface and functional spec

Consensus - Agreement on log and agreement on state. Abstract representation for state (represent storage) - interface for adding pending tx, and committing a block. Claiming tx-block. Build_block, advance block through change, handle disputes. Init, and clean-up.
Quorum voting - messages that drive internal state-machine, manages timing events (ripple-like - periodic kickoff of vote).
POET dynamic adaptation - targeting particular commit rate - sampling.
Quorum - 60%.
Timing-events - block-building. Tolerance for commit times.
Incentives - provided by transaction family.

Consensus Layer Interfaces:

Communication Layer (transport)

- Register handler for message-type

Business Logic / Smart Contract Layer
Storage Layer

TX validation logic - needs internal interface (double spend - need read-only interface, wrtite additional state). Similar to stored procedures in DB.
External - higher level application - work-flow.

1. Tx pre-filtering - is this a sensible tx, viable candidate
2. Candidate block with tx
3. Test that block against current state
4. Consensus process - vote on (tx on block are sane, new consistent state)

h