

FOSDEM 2020

The Confidential Consortium Framework

Amaury Chamayou



CCF: Multi-party applications



Verifiable consortium governance



Fine-grained confidentiality



Simple programming model

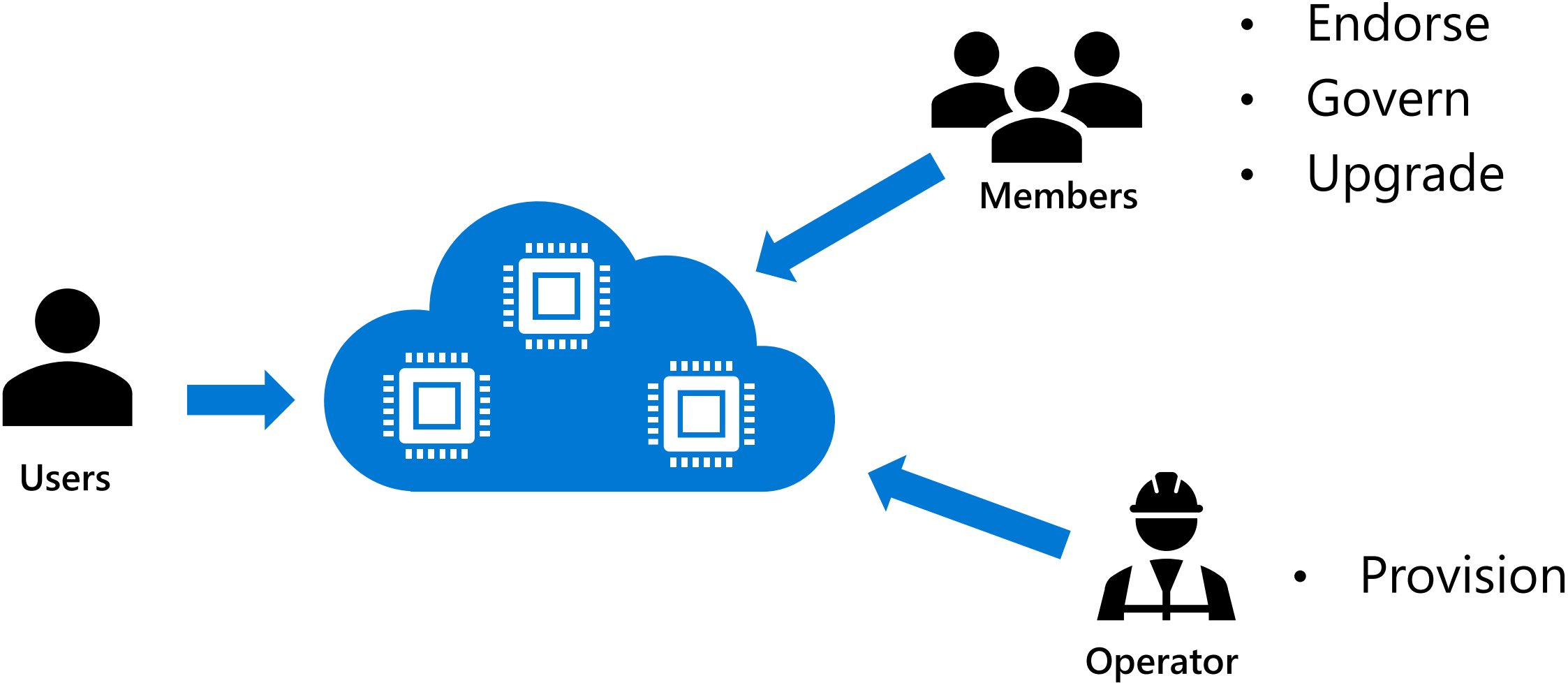


High availability



High efficiency

CCF: Multi-party applications



A network of Trusted Execution Environments

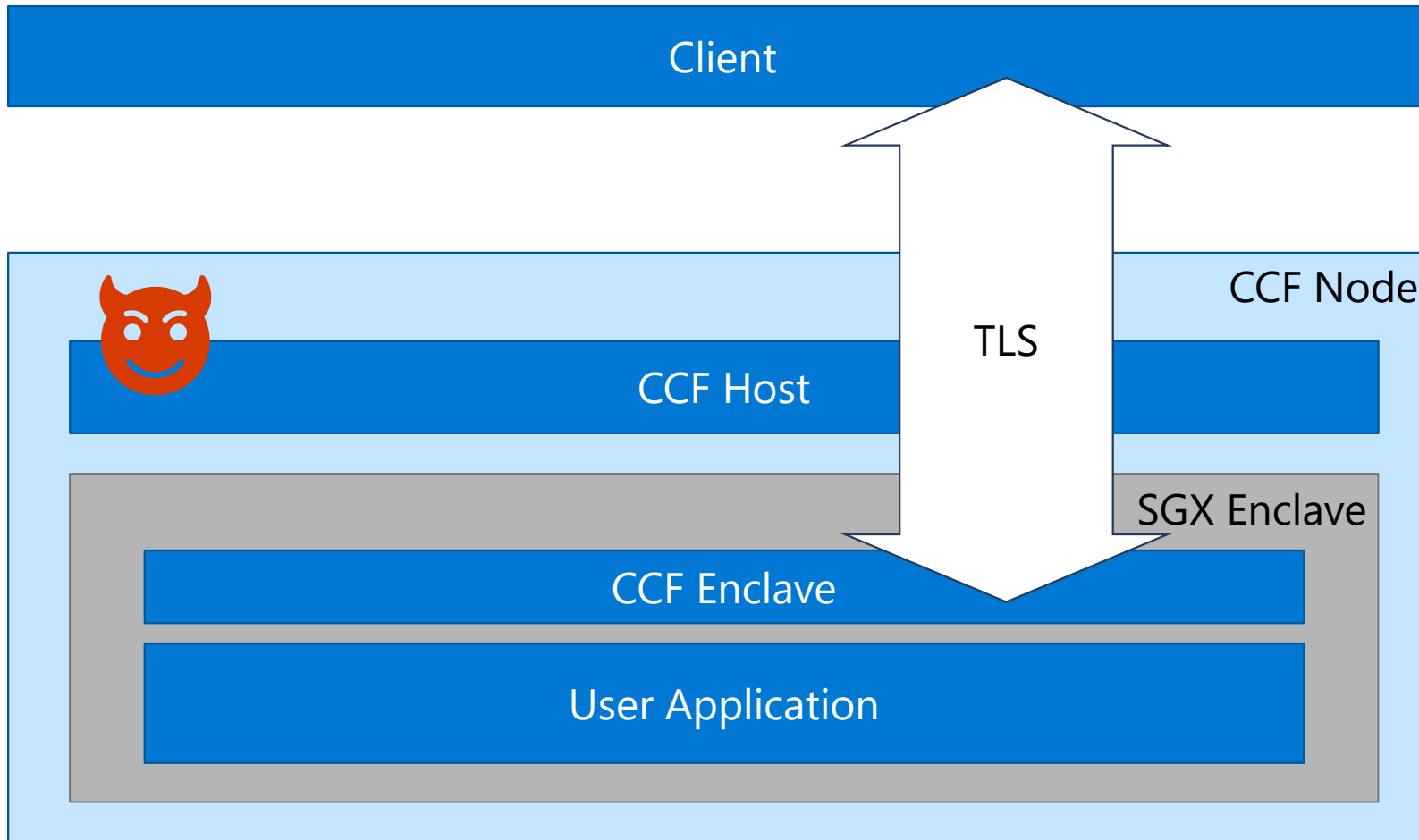
Trusted Execution Environments

- Encrypted and integrity-protected memory
- Cryptographic evidence over running code
- Remote attestation



Distributed
Trusted
Computation

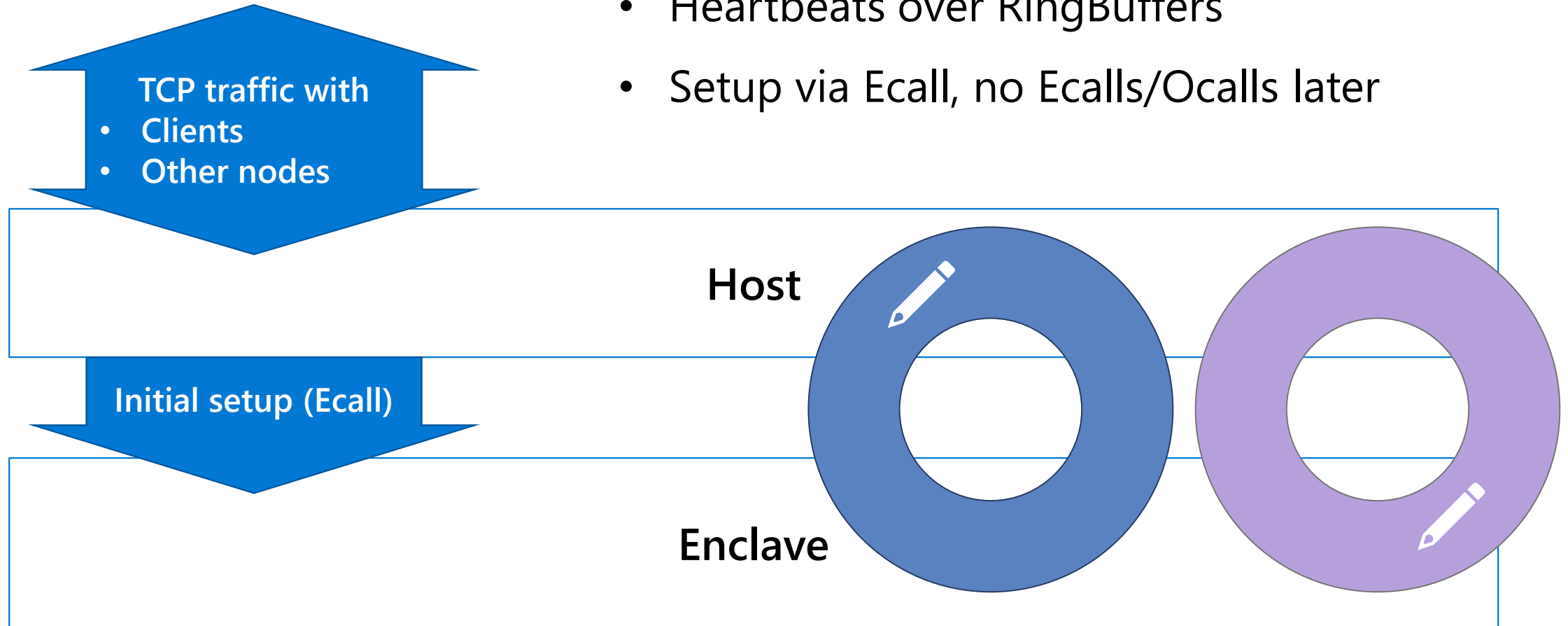
Node Overview



- TLS terminates in Enclave
- Host untrusted
- Enclave contains:
 - Application Logic/State
 - Governance
 - Fault Tolerance


Host-enclave communication

- TCP traffic forwarded via RingBuffers
- Heartbeats over RingBuffers
- Setup via Ecall, no Ecalls/Ocalls later



Join protocol

Adding a node to a CCF network

- Node
 - Create key pair
 - Send enclave quote to network
 - Platform
 - Code
 - Identity
- Network  Governance
 - Endorse identity
 - Send data secrets
- Node
 - Part of network
 - Catch up on state

Programmable, verifiable Governance

Governance

- Consortium of members
 - endorse initial ledger and configuration
- Stage votes
 - Membership
 - Users
 - Network Configuration
 - Code
 - Constitution
- Voting proposal are scripts
- Votes are scripts too!

Constitution sample

```
tables, calls, votes = ...

member_votes = 0

for member, vote in pairs(votes) do
  if vote then
    member_votes = member_votes + 1
  end
end

-- count active members
members_active = 0

tables["ccf.members"]:foreach(function(member, details)
  if details["status"] == STATE_ACTIVE then
    members_active = members_active + 1
  end
end)
```

```
-- check for raw_puts to sensitive tables
SENSITIVE_TABLES = {"ccf.whitelists", "ccf.gov_scripts"}
for _, call in pairs(calls) do
  if call.func == "raw_puts" then
    for _, sensitive_table in pairs(SENSITIVE_TABLES) do
      if call.args[sensitive_table] then
        -- require unanimity
        return member_votes == members_active
      end
    end
  end
end

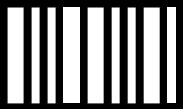
-- a majority of members can pass votes
if member_votes > math.floor(members_active / 2) then
  return true
end

return false
```

Proposal and Vote samples



```
tables, node_id = ...  
return Calls:call("new_user", user_cert)
```



```
tables, code_digest = ...  
return Calls:call("new_code", code_digest)
```



```
tables, changes = ...  
return (#changes == 1 and  
        changes[1].func == "new_code" and  
        changes[1].args[1] == NEW_CODE_DIGEST)
```

Code update

- Member vote to add new supported code version
- Members vote for new configuration
 - Add new nodes
 - Retire old nodes
- Members vote to remove old code version
- Constitution rules determines vote outcome

Recovery

- On loss of $> f$ nodes
- Back to original root of trust: members
 - Key shares
- New service
 - From old ledger
 - Endorsed by old ledger

Verifiability

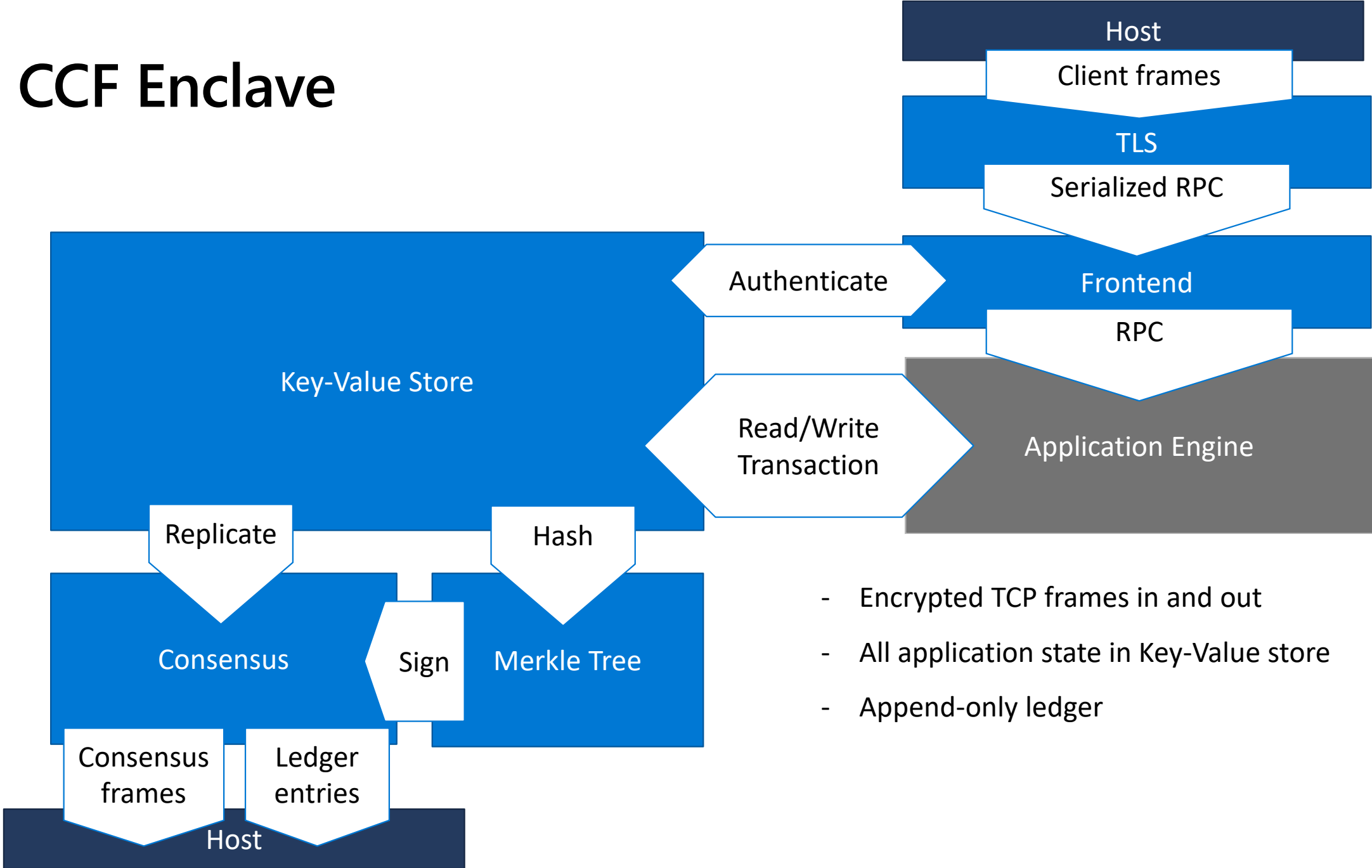
- Governance state is public
- Governance transactions are signed
- Same total order as other transactions
- Tamper-proof ledger

A simple programming model

Data in CCF is...

- Encrypted at rest
 - In the ledger
- Encrypted in motion
 - On the wire during replication
- Encrypted during computation
 - Enclave memory is encrypted during execution

CCF Enclave



- Encrypted TCP frames in and out
- All application state in Key-Value store
- Append-only ledger

Consensus

- Deterministic commit
- Crash-fault tolerance
 - In-enclave Raft variant
 - Robust to f out of $2f + 1$ failures
 - Enables blaming compromised nodes
 - Relies on TEE for confidentiality and integrity
- Byzantine fault tolerance
 - In-enclave PBFT variant, work in progress
 - Robust to f out of $3f + 1$ simultaneous malicious nodes
 - Relies on TEE for confidentiality

Key-value Store

- Key-Value Maps
 - `get(key)`
 - `put(key, value)`
- Transactions
 - Strict serializability
 - Opacity
- App-driven confidentiality
 - Arbitrary reveal
- Code in Store
 - Scripting runtimes

Transaction receipts

- Merkle Tree paths
- Self-verifying
- Signed by service
- Offline proof/verification

CCF Apps can be written in...

- Native
 - C++
- Runtimes with code stored in KV
 - Lua
 - EVM languages (Solidity...)
 - JavaScript/ES2015

The Confidential Consortium Framework

github.com/microsoft/CCF

