DSR

DOING SOFTWARE RIGHT

# ZB Ledger

## Agenda

- Project Scope and Main Goal
- Use Cases
- Current Status
- Next Steps
- Network topology
- Authentication and authorization
- Model Info Use Case
- About Tendermint
- Appendix
    - Assumptions

## Project Scope and Main Goal

- Create a Public Permissioned Distributed Ledger owned and hosted by ZB Alliance members
- The Ledger can store information about Device Models (created by Manufacturers) and Compliance tests (created by Testers and Alliances)
- Write access to the Ledger is permissioned and restricted
- Anyone can read from the Ledger

## Use Cases: Write Data to the Ledger

- **Use Case 1: Writing Device Model Info by Manufacturer**
    - As a Manufacturer, I need to write information about a Device Model, Firmware and Hardware to the Ledger, so that it can be read by anyone from the ledger for compliance and other purposes
- **Use Case 2: Writing a Certificate by Manufacturer**
    - As a Manufacturer, I need to write Device's a certificate to the Ledger, so that it can be read by anyone from the ledger for PKI use cases (for example, when adding a new device to the Network)
- **Use Case 3: Writing Compliance Tests results by Testing Organization**
    - As a Compliance Tester, I need to write result of compliance tests for each tested Device Model to the Ledger, so that it can be read by anyone from the ledger for confirming compliance test results and other purposes
- **Use Case 4: Writing Compliance Confirmation by ZB**
    - As an Alliance (ZB Alliance for example), I need to write confirmation of the Compliance Test Results to the Ledger, so that it can be read by anyone from the ledger for compliance checks (for example, when adding a new device to the Network)

## Use Cases: Read Data from the Ledger

- **Use Case 5: Check for Device Compliance**
  - As a ZB (or any other) Network, I need to know if the Device is compliant with the ZB (or other) standard by reading information from the Ledger, so that I can add it to the Network
- **Use Case 6: Reading Device Model Info**
  - As a ZB (or any other) Network, I need to know Device's Model Info including Firmware and Hardware versions by reading information from the Ledger
- **Use Case 7: PKI**
  - As a user, I need to get a chain of X509 certificates to verify the authority of the given X509 certificate.

## Current Status

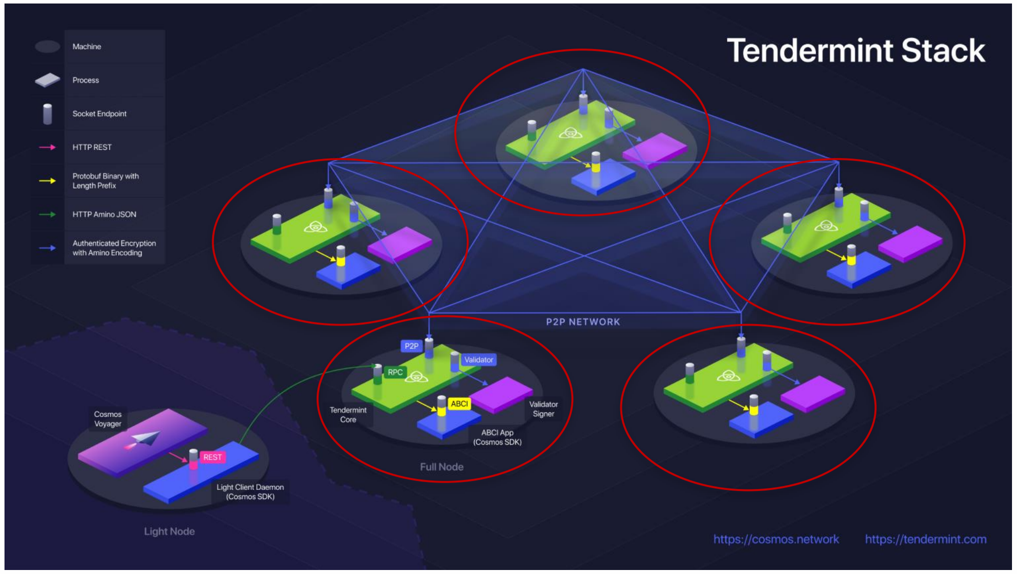
- The Ledger is up and running
- Core logic is implemented on top of Tendermint and Cosmos SDK
- All main use cases are basically implemented:
- There is a UI available
  - Can be used to browse the ledger
  - Can be use to send new transactions but only for Demo use cases (as the signing keys are stored on the backend)

## Next Steps

- Finish implementation
  - IP correlation problem in particular
- Make the code public in GitHub
- Pilot Release: September 2020

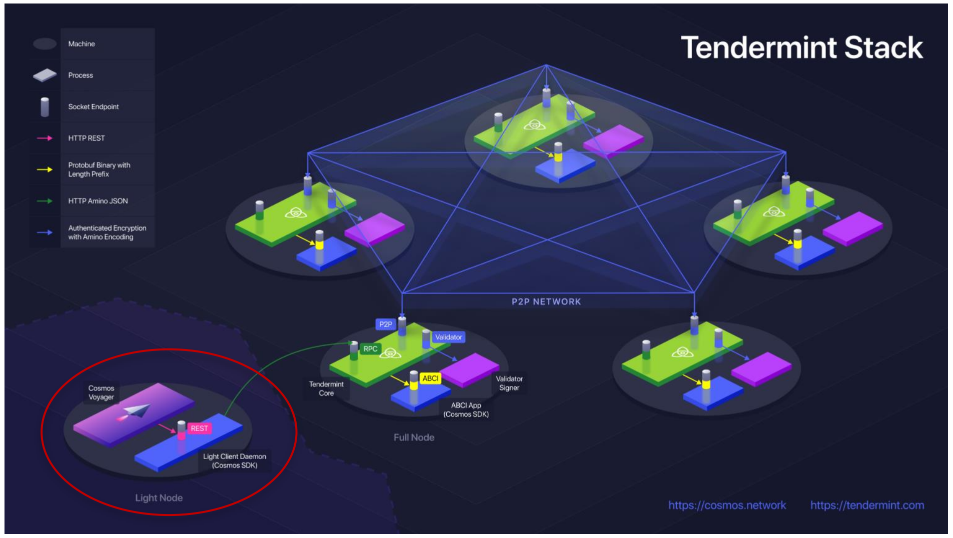
# Network topology: Validator Nodes

- Come to consensus
- Maintain ledger
- Permissioned
- Not anyone can be a Validator Node
- Number of nodes can be extended



# Network topology: Clients

- Anyone writing or reading
- Client != Validator
- Don't need to maintain or have a full Ledger
- Can read from 1 Node only
- Write access to the Ledger is restricted (write permission needs to be granted)
- Read access to the Ledger is public (anyone can read)



## Clients

- CLI
  - Run on a user’s machine
- REST API
  - Run as a server: either a local deployment per organization/application/user, or a global one (current UI)
- Client Libs / API Used by the application itself
  - Core API is there, but language and platform-specific libraries - TBD

## Network topology

- Currently there are 4 Validator Nodes in AWS
  - Different regions
  - 3 Nodes owned by DSR, 1 Node owned by Comcast
- There is a backend and web UI which can be used as a client

## Authentication

- Every transaction (write request) must be signed
- No signatures/authentications for read requests
- Sender must have an Account on the Ledger (as a transaction)
- The public key used for signature verification must be on the Ledger (associated with the Account transaction)

## Authorization

- Role-based authorization
- Roles:
  - Admin:
    - Can assign or revoke Manufacturer role to an Account
    - Can create new Accounts (TBD)
  - Manufacturer:
    - Can create, edit or delete Model Info
- More roles will be created for other use cases

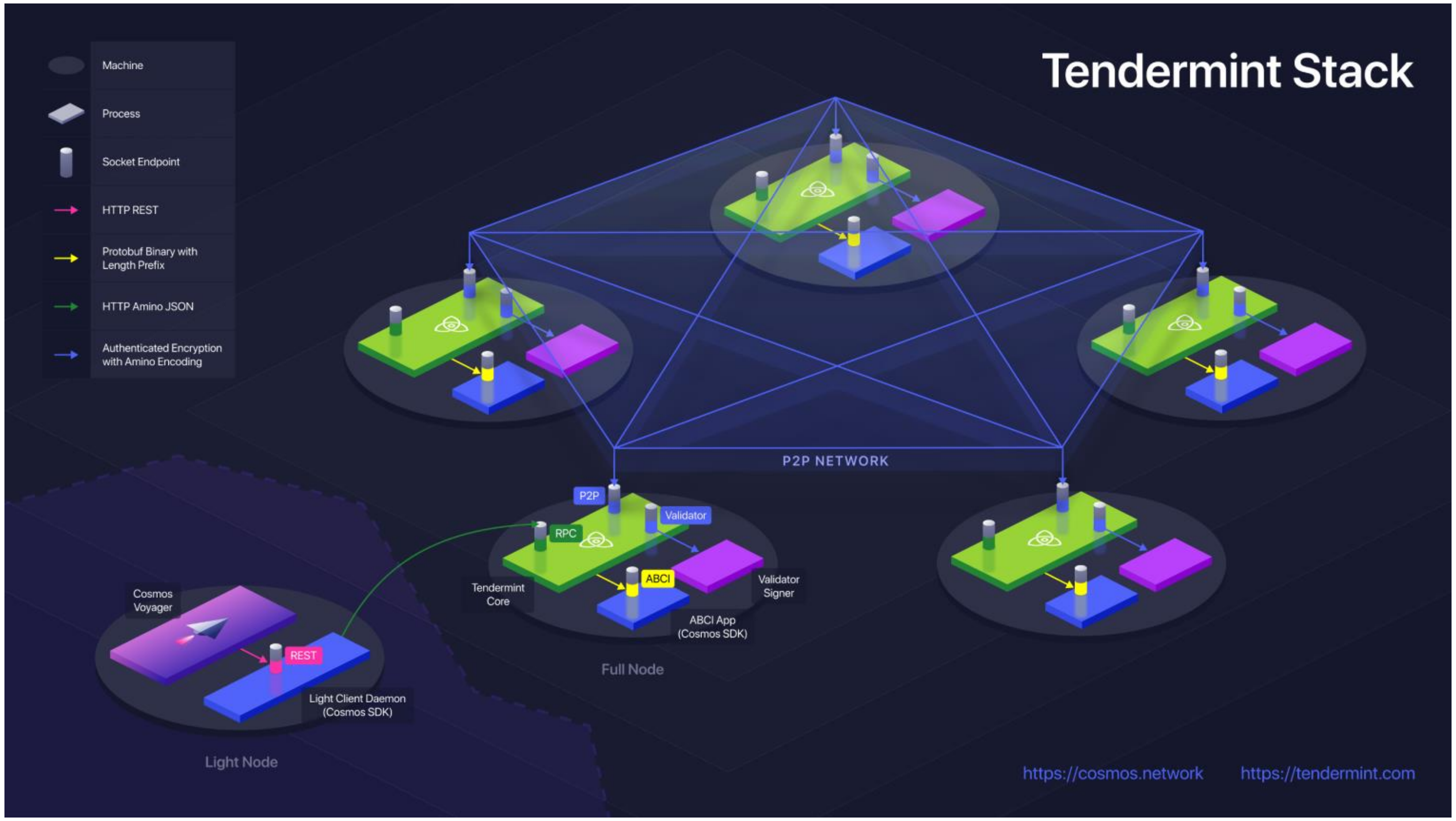
## About Tendermint

- Tendermint is a blockchain application platform performing Byzantine Fault Tolerant (BFT) State Machine Replication (SMR) for arbitrary deterministic, finite state machines.
- It  provides the equivalent of a web-server, database, and supporting libraries for blockchain applications written in any programming language. Like a web-server serving web applications, Tendermint serves blockchain applications.

## About Tendermint

- https://github.com/tendermint/tendermint/
- Apache 2.0
- Big community
- 3.5K stars on GitHub
- Projects in Production

# About Tendermint



## About Tendermint: BFT

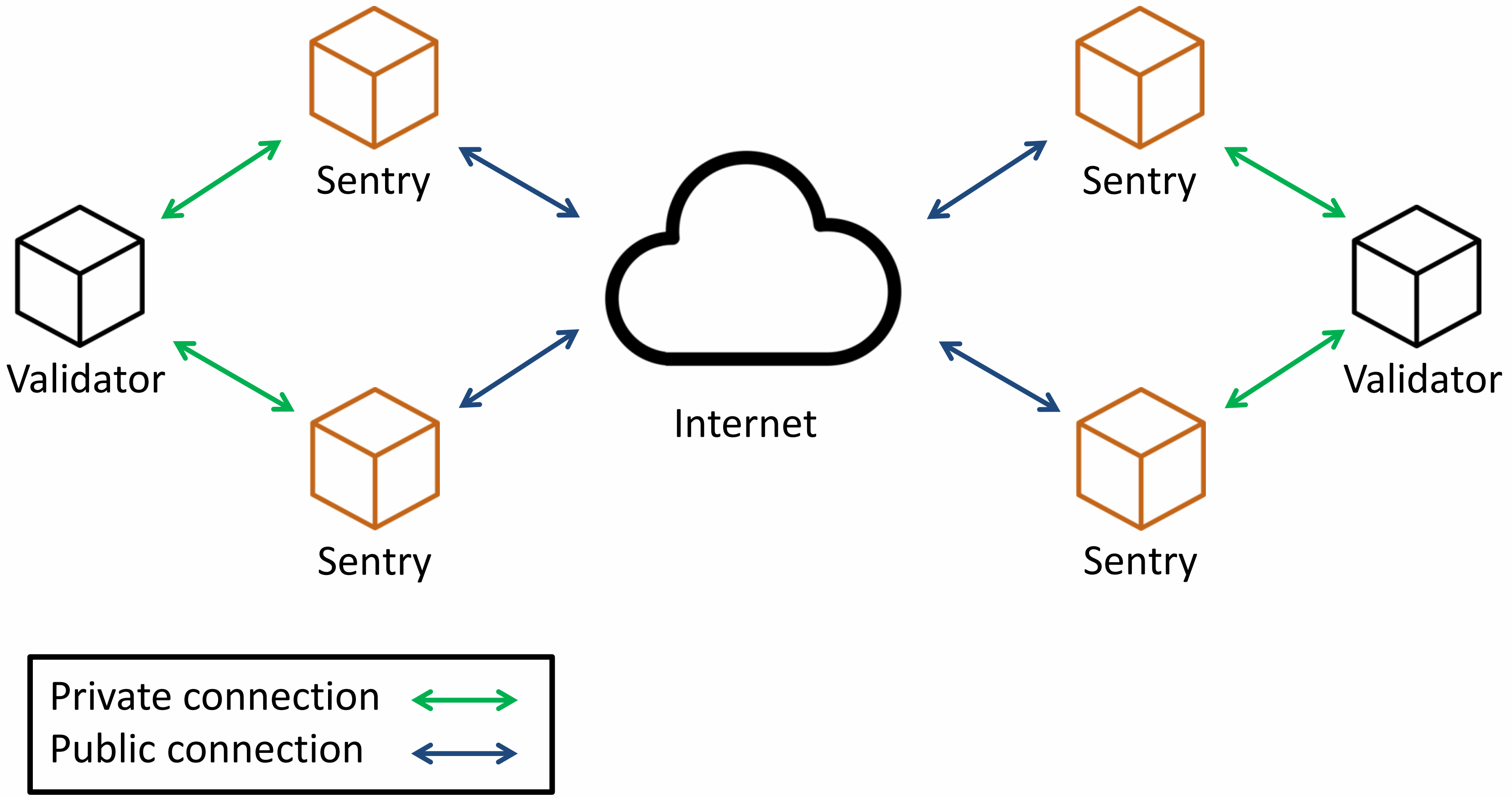
- Tendermint works even if up to 1/3 of machines fail in arbitrary ways (either fail-stop/crash, or behave maliciously)
  - 1 Node in the current ZB Ledger may crash, and the network will still be working
  - 1 Node in the current ZB Ledger may behave maliciously (for example, ignore authentication rules), and the network will still be correctly working (for example, enforce authentication rules)

## About Tendermint: Light Clients

- Light client can read state from any **single** node without downloading full transaction log and trust the result
- Don’t need to have/maintain a full Ledger on the client side
- This is achieved using special data structure for state storage
  - Allows proving that given data is present or absent in the state
  - IAVL in Tendermint
  - Patricia Merkle Tree in Etherium

## About Tendermint: Sentry Nodes

DDoS mitigation strategy

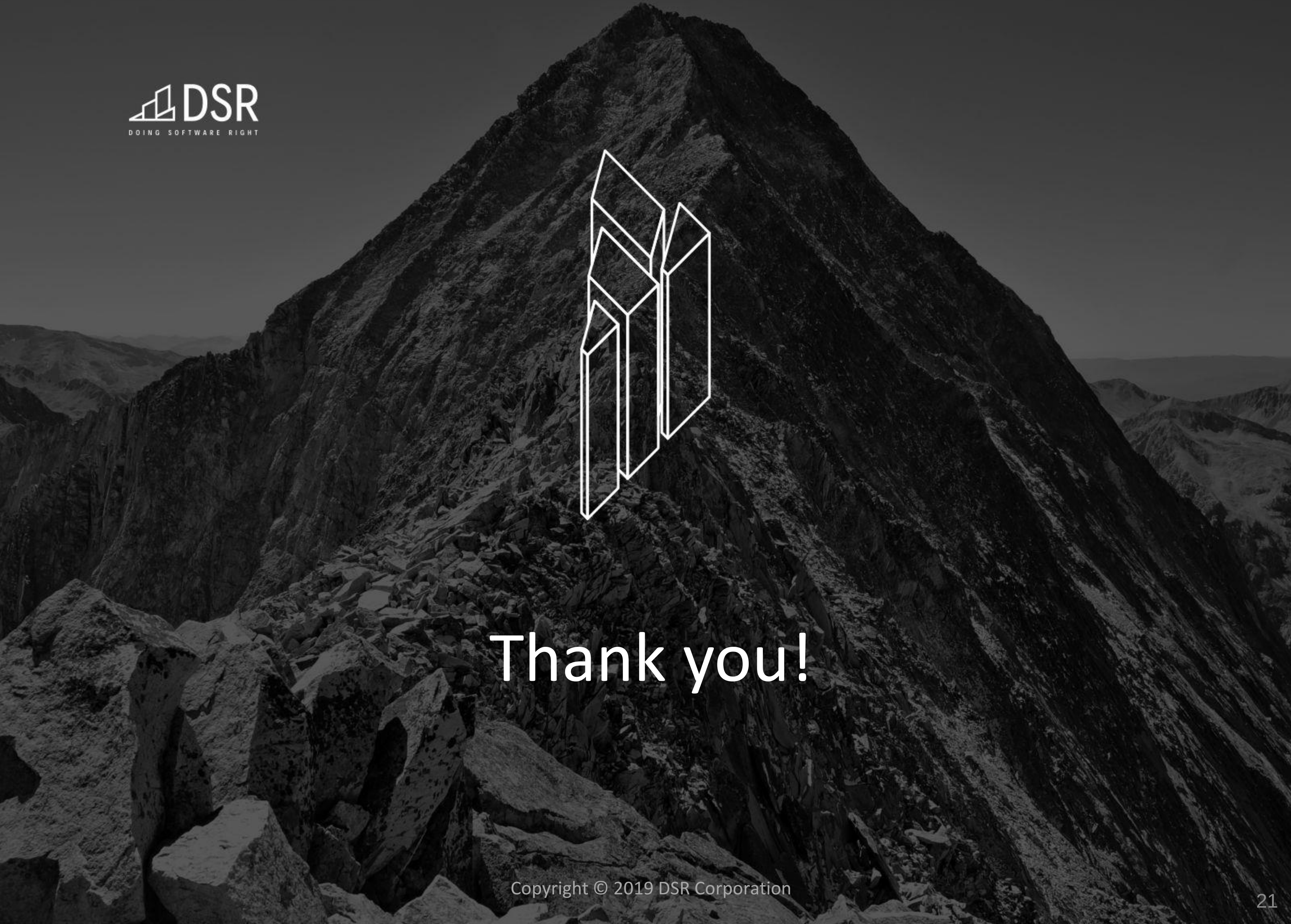


## APPENDIX: Assumptions

- Pool
  - The Ledger is maintained by a number of Nodes belonging to ZB Alliance members
  - The Nodes can be Validators (participating in Consensus) and Observers (replicating the Ledger and used for read access)
  - The number of Validator Nodes is expected to be greater than 4 but less than 25
- Access
  - Write access to the Ledger is restricted (write permission needs to be granted)
  - Read access to the Ledger is public (anyone can read)
- Data stored on the Ledger
  - Individual Devices are not stored on the Ledger, only Device Models are stored
  - Expected number of Device Models to be stored is equal to the current number of certified devices

DSR

DOING SOFTWARE RIGHT



## Thank you!