Chainstack

Quorum meetup

How do private transactions work on Quorum?

Syahrul Nizam, Developer Advocate

Presented by

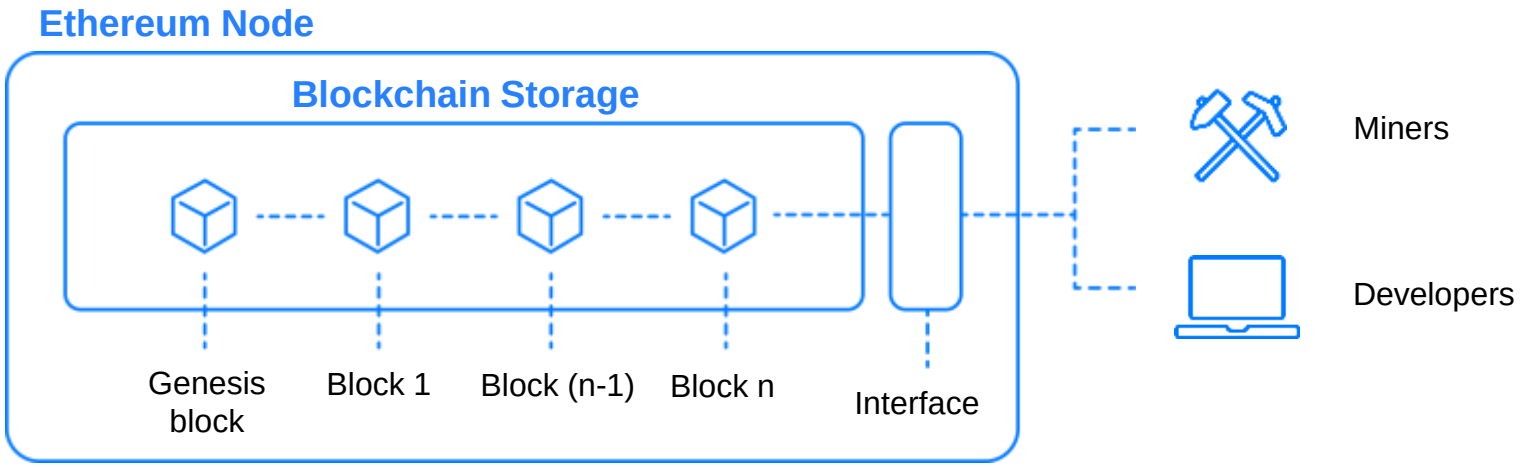
Syahrul Nizam

Developer Advocate

Blockchain enthusiast with experience in creating & deploying Smart Contracts on Ethereum

The innerworkings of Ethereum

The Ethereum network is made up of nodes, each running an Ethereum client like Geth

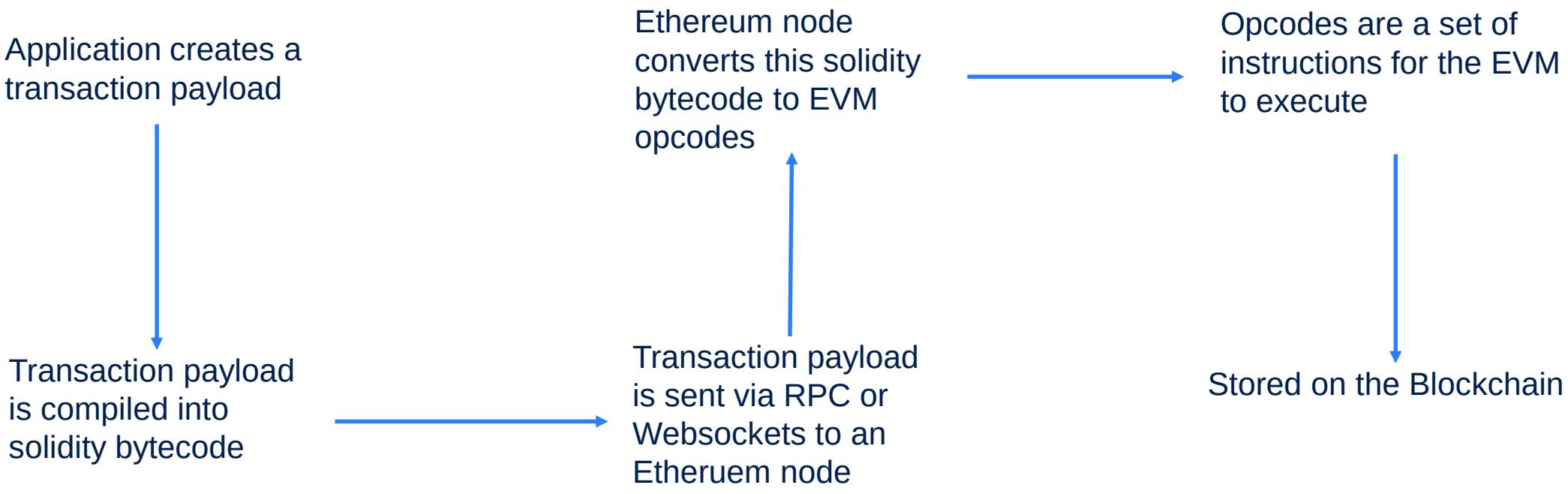


What is Quorum?

1. Quorum is a Ethereum based blockchain, designed for use by Enterprises.
2. It provides smart contract capability, much like Ethereum
3. What makes Quorum special is that it has support for private transactions, which is impossible on Ethereum (somewhat)

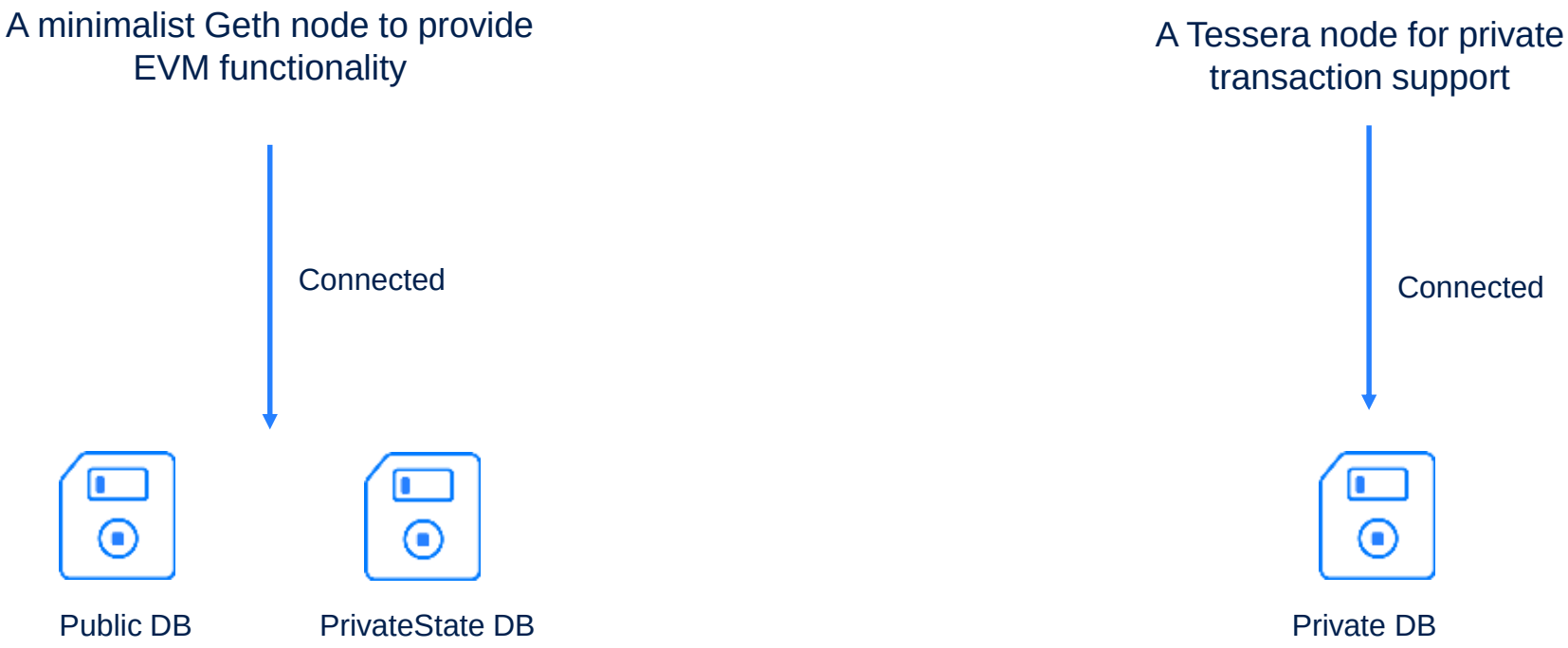
Transactions in Ethereum

A graphical representation of how transactions are propagated in a regular Ethereum network



Quorum nodes

A quorum node consists of two things….



Let’s visualize the private transaction

Application Layer

Quorum transaction payloads are identical to Ethereum’s transaction payloads with an additional privateFor parameter

Alice wishes to make a private transaction to Bob

She supplies Bob’s Tessera public key in the privateFor parameter of the transaction payload:

```
let txPayload = {
    from:address,                    //Ethereum
    to:contractAddress,              //Ethereum
    gasLimit:4000000,                //Ethereum
    data:encodedABI,                 //Ethereum
    privateFor: [tesseraPublicKey]   //Quorum
  }
```

Sending the transaction payload

Application Layer

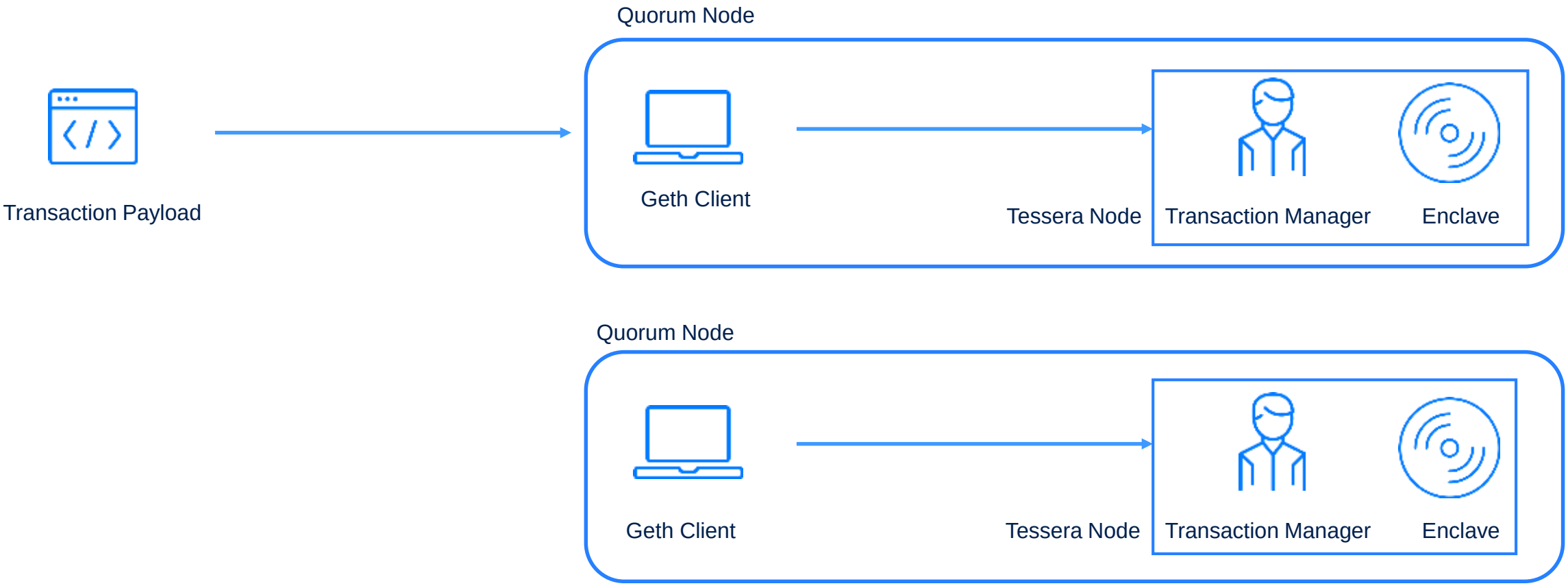
The application then sends the transaction payload to **Alice's** Quorum node

```
axios.post(quroumRPC,{
        "jsonrpc":"2.0",
        "method":"eth_sendTransaction",
        "params":[txPayload]
        "id":1
    }).then((response) => {
        return response.data.result
    })
```

Quorum node

The Quorum node detects the presence of the privateFor parameter in the transaction payload.

Each Quorum node comes with it’s own paired Tessera Node (Transaction Manager + Enclave)



Tessera node (Alice's)

Enclave

Transaction Manager

Transaction manager makes a call to the Enclave with the transaction payload

Enclave

Enclave verifies the transaction payload

Enclave generates a random nonce(an arbitrary number) and a RMK (an arbitrary password)

```
nonce = 0
RMK   = "92asd321aasd..."
```

Tessera node (Alice's)

Enclave

```
let txPayload = {
    from:address,                   //Ethereum
    to:contractAddress,             //Ethereum
    gasLimit:4000000,               //Ethereum
    data:encodedABI,                //Ethereum
    privateFor: [tesseraPublicKey]  //Quorum
  }
```

EncodedABI payload is encrypted using the nonce and RMK

```
encryptedTxPayload = "p93xan+)xj2m9_xa..."
```

Tessera node (Alice's)

Enclave

```
nonce = 0
RMK   = "92asd321aasd..."
```

Alice's Tessera Private Key and Bob's Tessera Public Key (supplied in transaction payload) and a random Number is used to encrypt the RMK itself

```
encryptedRMK = "1am-0ai094das"
```

Transaction Manager

Encrypted RMK and encrypted payload sent back to Alice's TM

Tessera Node (Alice's)

Transaction Manager

The transaction manager (Alice) hashes the encrypted payload, and uses the hash as a key to store the encrypted payload & encrypted RMK in the **private database**

```
encryptedTxPayload = "p93xan+)xj2m9_xa..."
encryptedRMK = "1am-0ai094das"
hash = "0x98433208342"
```

A	B	C
Key	EncryptedRMK	EncryptedPayload
0x98433208342	1am-0ai094das	p93xan+)xj2m9_xa...

This same information is sent to Bob's transaction manager and stores in his own **private database**

Bob's Transaction Manager

Bob's Transaction manager acknowledges receipt

Alice's transaction manager then returns hash to Alice's Quorum node

Quorum node (Alice's)

Alice's Quorum node then replaces the **data** parameter in the transaction payload with the hash received from Alice's transaction manager

```
let txPayload = {
    from:address,                   //Ethereum
    to:contractAddress,             //Ethereum
    gasLimit:4000000,               //Ethereum
    data:encodedABI,                //Ethereum
    privateFor: [tesseraPublicKey]  //Quorum
  }
```

Note the change in the data parameter

```
let txPayload = {
    from:address,                   //Ethereum
    to:contractAddress,             //Ethereum
    gasLimit:4000000,               //Ethereum
    data:"0x98433208342",           //Replaced
    privateFor: [tesseraPublicKey]  //Quorum
  }
```

It also replaces the **v** value in the transactional receipt to tell other quorum nodes that it is a private transaction, and not a normal transaction with nonsensical bytecode

Now the transaction can be propagated by the node into the blockchain normally as per standard Ethereum protocol. Every quorum node will receive this private transaction.

Quorum network

1. All parties will receive the private transactions
2. Everyone will try to decrypt it

How do nodes know that they are party to the private transaction??

Quorum Node (Bob's)

Bob is party to the private transaction

Quorum Node

Bob's quorum node receives a transaction receipt with a **V** value of 37 or 38.

```
hash = "0x98433208342"
```

Bob's quorum node then obtains the hash in the **data** parameter of the transaction payload

Transaction Manager

Bob's transaction manager will do a lookup for this hash and see if it exists!

A	B	C
Key	EncryptedRMK	EncryptedPayload
0x98433208342	1am-0ai094das	p93xan+)xj2m9_xa...

Tessera Node (Bob's)

```
encryptedTxPayload = "p93xan+)xj2m9_xa..."
encryptedRMK = "1am-0ai094das"
```

Since Bob holds the transaction, he can obtain the encrypted payload and encrypted RMK in his private database

Using his Tessera Private Key, he can first unencrypt the encrypted RMK

```
encryptedTxPayload = "p93xan+)xj2m9_xa..."
RMK    = "92asd321aasd..."
```

Using this unencrypted RMK, he can now decrypt the encrypted transaction payload

```
let txPayload = {
    from:address,                    //Ethereum
    to:contractAddress,              //Ethereum
    gasLimit:4000000,                //Ethereum
    data:encodedABI,                 //Ethereum
    privateFor: [tesseraPublicKey]   //Quorum
  }
```

Bytecode sent back to Bob's transaction manager

The decrypted payload is valid bytecode

Quorum Node (Bob's)

Using the unencrypted bytecode, the EVM can now do computations as per normal.

Bytecode is discarded once it's used and **privateStateDB** is updated accordingly to the instructions in the opcodes

Quorum Nodes (Candice)

Candice is not party to the transaction sent by Alice

Candice's quorum node receives a transaction payload with a **V** value of 37 or 38

Candice's quorum node then obtains the hash in the **data** parameter of the transaction payload

Candice's transaction manager will look up this hash, and the corresponding encrypted data does not exist.

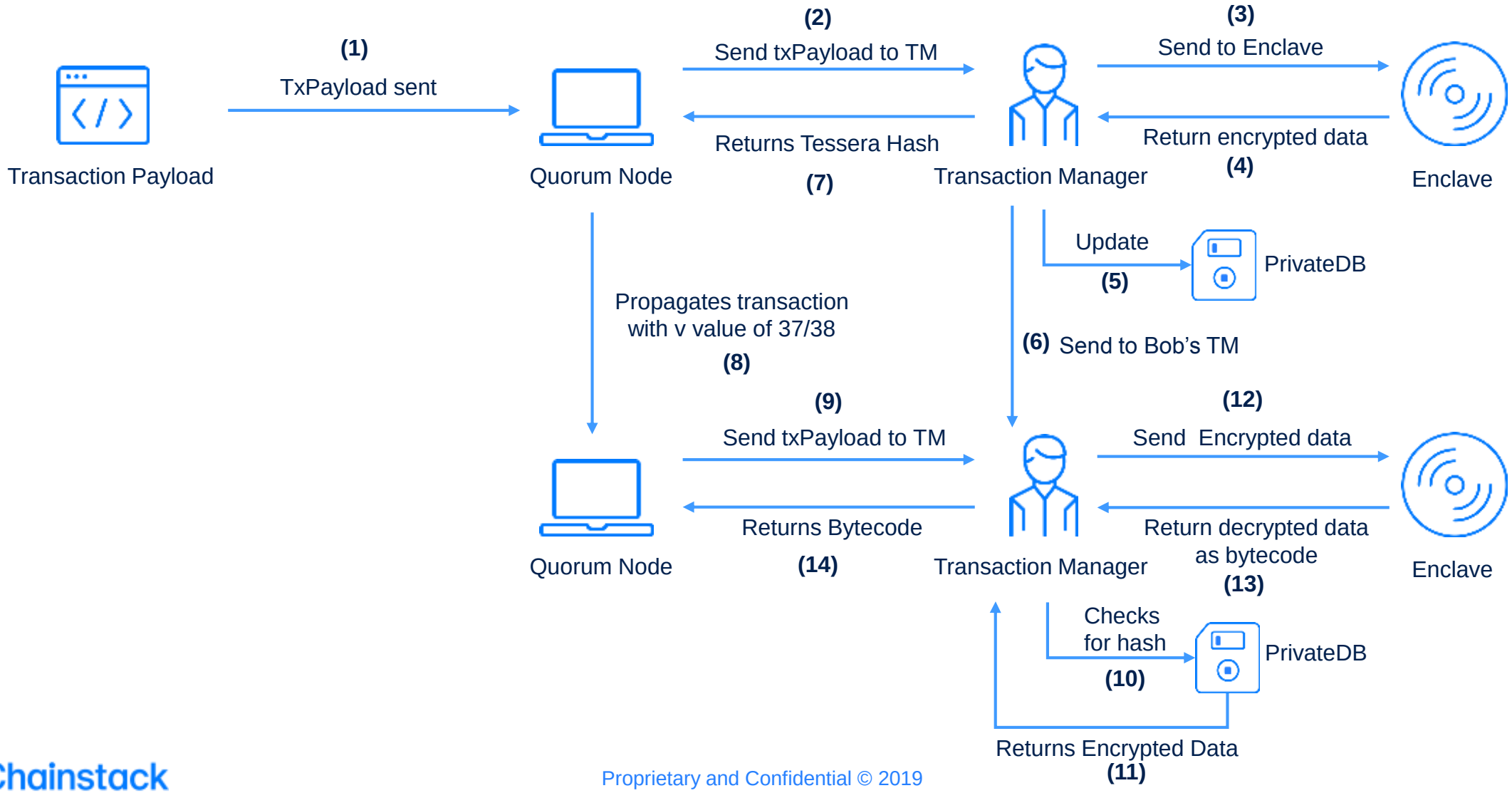
Candice's node skips this transaction.

Demonstration & Code

Let’s spin up a Quorum network using Chainstack

1. Create a project on Chainstack
2. Download this repository : https://github.com/chainstack/quorum-iot-tutorial
3. Install Node.JS
4. Enjoy!

Summary



What’s next?

Get started with Chainstack

1. Visit our documentation site to quickly create your own Dapp platforms
   https://docs.chainstack.com
2. Create an account on Chainstack (2 week trial)
   https://console.chainstack.com
3. Have fun!

Q & A

Contact

syahrul.nizam@chainstack.com

Helpful links

https://docs.chainstack.com

https://console.chainstack.com