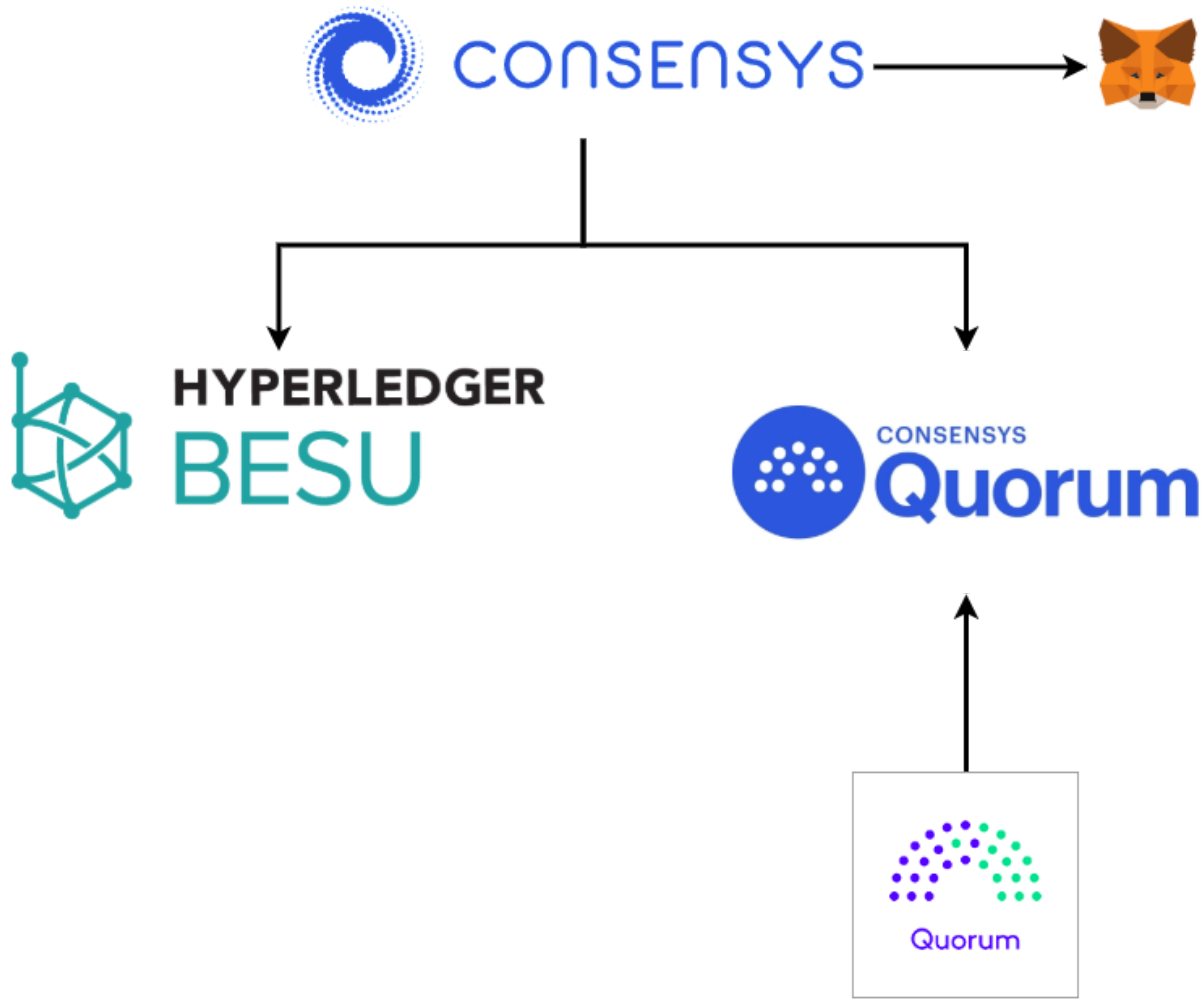
22/03/2021 status update

From BPMN to Permissioned blockchains: A model-driven approach.

Nizar Hmain

NEW YORK, NY, Aug. 25, 2020—ConsenSys, an industry-leading blockchain software company, today announced the acquisition of Quorum®, an enterprise-variant of the Ethereum blockchain developed by J.P. Morgan (NYSE: JPM). With the addition of Quorum, ConsenSys now offers a full range of products, services, and support for Quorum, accelerating the availability of features and capabilities—such as digital asset functionality and document management.


CONSENSYS
HYPERLEDGER
BESU
CONSENSYS
Quorum
Quorum

nizarhmain / ChorChain

forked from am1987/ChorChain

Default branch

`master` Updated 4 days ago by nizarhmain

Your branches

`besu` Updated 3 days ago by nizarhmain

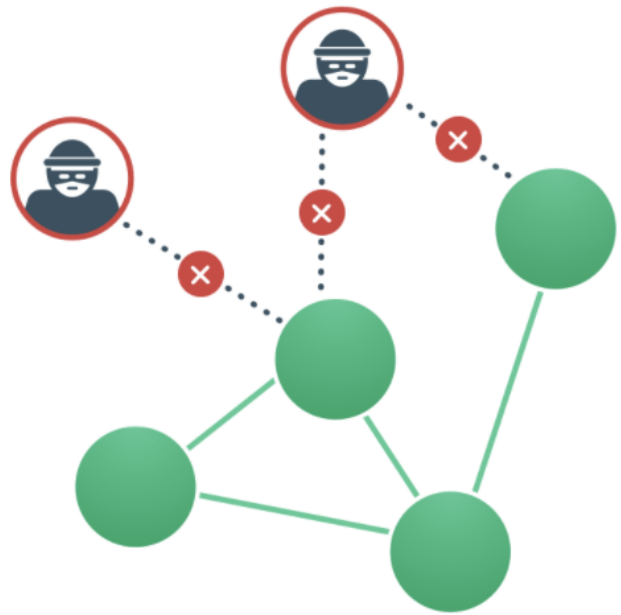
Active branches

`besu` Updated 3 days ago by nizarhmain

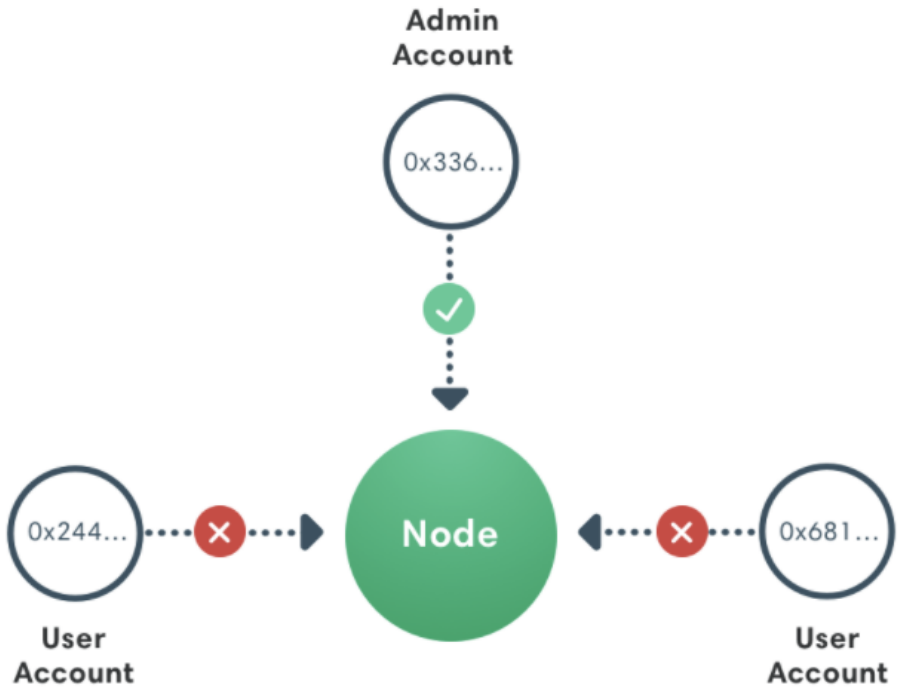
`Multi-chain` Updated 26 days ago by AlessandroMarcellettiUnicam

Stale branches

`newTranslator` Updated 14 months ago by AlessandroMarcellettiUnicam

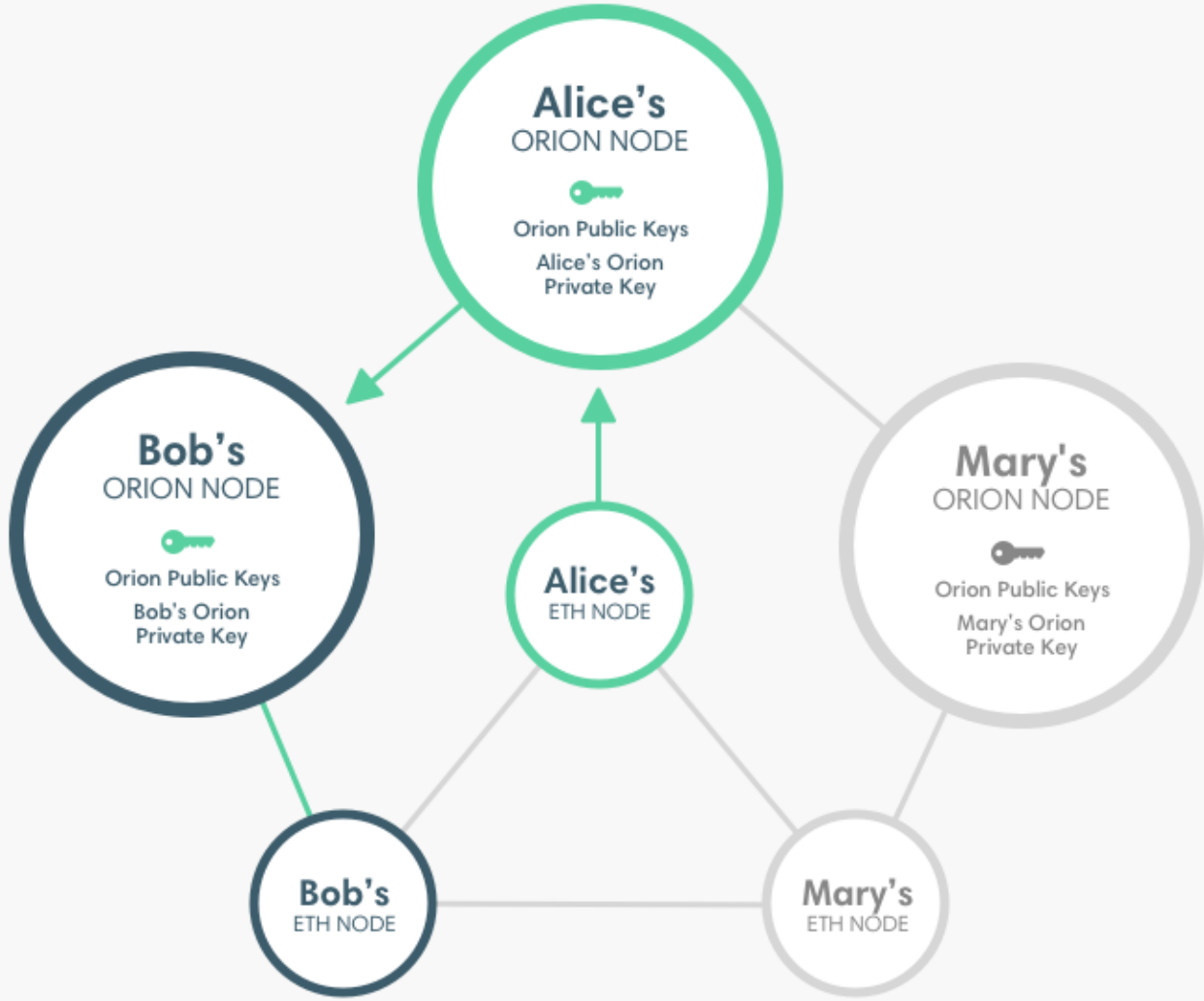


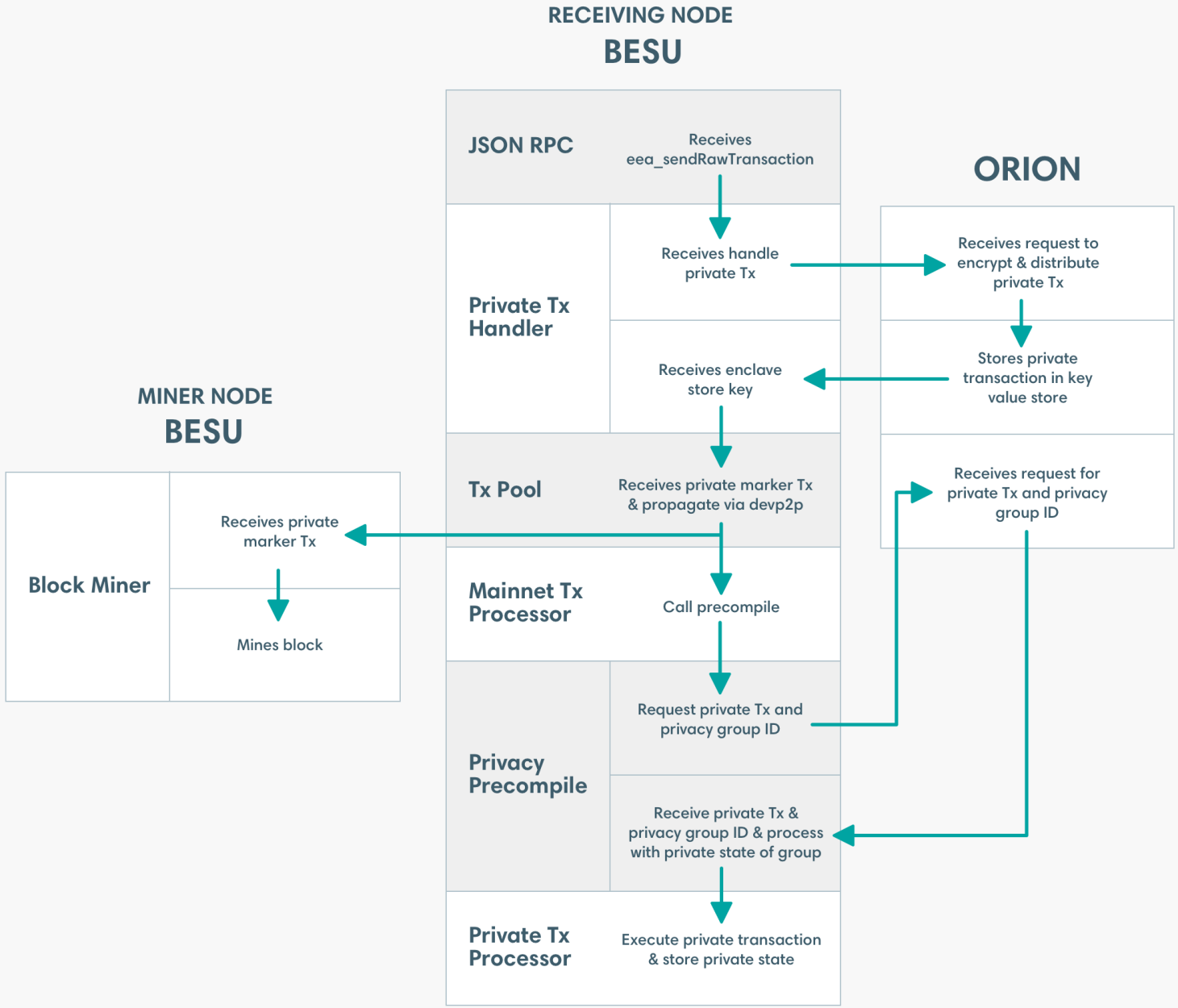
Node-level permissions are a useful system of governance to control connections to an individual node.



Using account permissions in Hyperledger Besu, a consortium blockchain can limit which accounts a node accepts transactions from, and which it rejects.

Alice sends a private transaction to Bob.

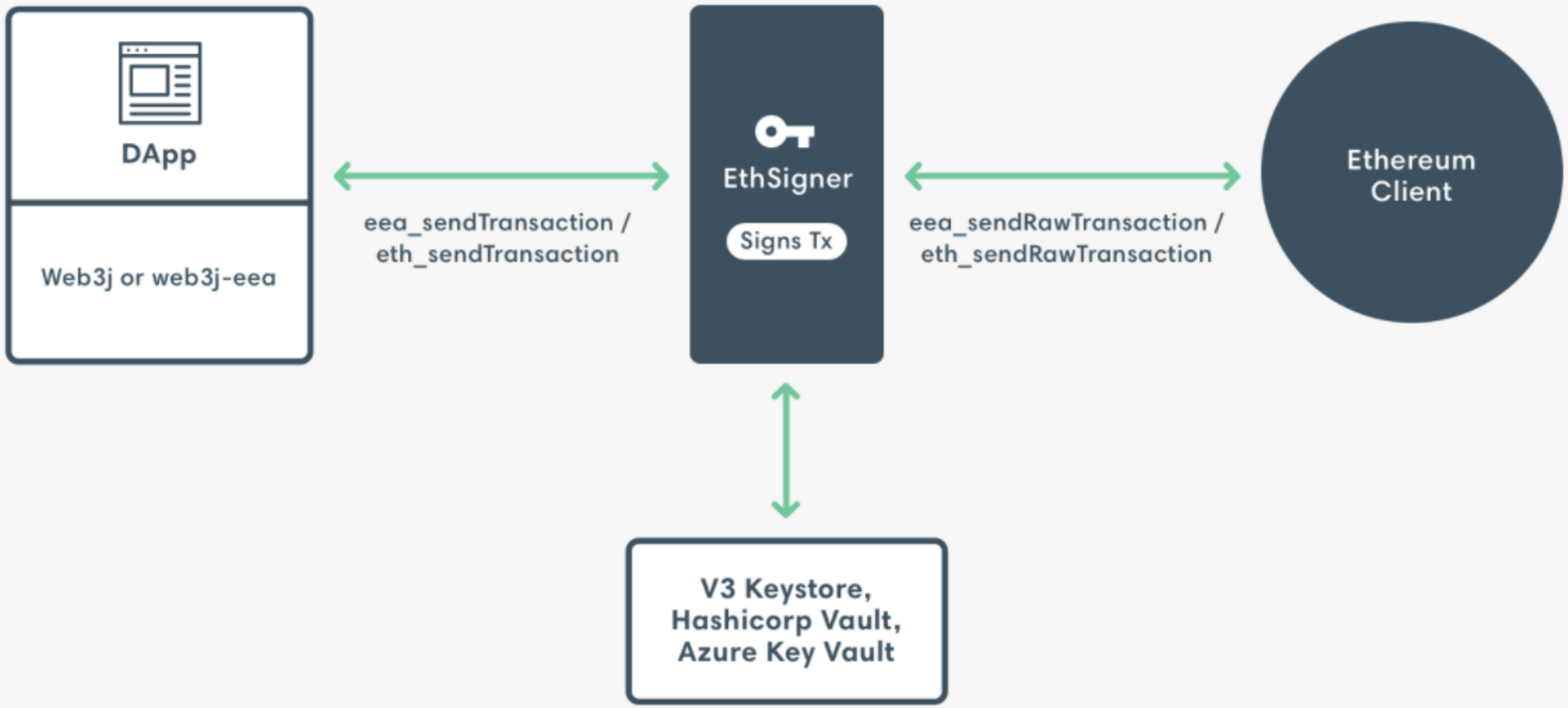



RECEIVING NODE
BESU
JSON RPC
Receives eea_sendRawTransaction
Private Tx Handler
Receives handle private Tx
Receives enclave store key
Tx Pool
Receives private marker Tx & propagate via devp2p
Mainnet Tx Processor
Call precompile
Privacy Precompile
Request private Tx and privacy group ID
Receive private Tx & privacy group ID & process with private state of group
Private Tx Processor
Execute private transaction & store private state
ORION
Receives request to encrypt & distribute private Tx
Stores private transaction in key value store
Receives request for private Tx and privacy group ID
MINER NODE
BESU
Block Miner
Receives private marker Tx
Mines block

EthSigner acts as a proxy service by forwarding requests to the Ethereum client. When EthSigner receives a transaction it generates a signature using the stored private key, and forwards the signed transaction to the Ethereum client.

Note: Besu does not implement `eth_sendTransaction()`, we must use `eth_sendRawTransaction` and sign it ourself or instead call `localhost:8545` the default rpc node, we call the ethSigner node that signs it for us `localhost:18545` with `eth_sendTransaction`

EthSigner Transaction



```
String rpc_endpoint = "http://localhost:8545";
String signer_proxy = "http://localhost:18545";
```

```
Web3j web3j = Web3j.build(new HttpService("http://localhost:7545"));
Admin adm = Admin.build(new HttpService("http://localhost:7545"));
Web3j web3j = Web3j.build(new HttpService(rpc_endpoint));
Web3j ethsigner = Web3j.build(new HttpService(signer_proxy));
Admin adm = Admin.build(new HttpService(rpc_endpoint));
```

```
//send sync
EthSendTransaction transactionResponse = web3j.ethSendTransaction(transaction1).send();
EthSendTransaction transactionResponse = ethsigner.ethSendTransaction(transaction1).send();
```

In order to send private transactions however, we need to use `eea_sendRawTransaction` instead of `eth_sendTransaction` which generates a privacy marker transaction and submits it to the tx pool, and returns the transaction hash of the privacy marker transaction. Essentially storing some information publically and some privately.

here comes the hart part. This method would not throw an exception when called and would just exit.

```
PrivateTransactionEncoder.signMessage(privTrac,  orion1_priv_key);

final String signedTransactionData =
                Numeric.toHexString(
                                PrivateTransactionEncoder.signMessage(privTrac, 2018, orion1_priv_key));

  //send sync
 final String transactionHash = eea.eeaSendRawTransaction(signedTransactionData).send().getTransactionHash();
```


7 HOURS
LATER...

The `web3j.eea` package needs v4.8.4
The `web3j.besu` uses as well v4.8.4
The `web3j.org` uses as well v4.8.4

I realized that maybe the signing method in utils was wrong .... or just the wrong version

updated `web3j.utils` `web3j.crypto` from version 4.1.1 to version 4.8.4 and it worked

Still need to cleanup and fix the interface to select nodes that would have access to the private smart contract and so on ...

1. Run ChorChain ✅
2. Besu public transactions ✅
3. Besu private transactions ✅ (ni)
4. GoQuorum public tx ❌
5. GoQuorum private tx ❌
6. Interface and code cleanup ❌