BLOCKCHAIN APPROACH TO CYBER SECURITY VULNERABILITIES

- KARAN VISHAL DUA 20BAI1012
- CHINTHAMANI MOHAN KRISHNA 20BAI1269
- PRANAV BHANOT 20BAI1087

Hyperledger fabric blockchain in cybersecurity vulnerability

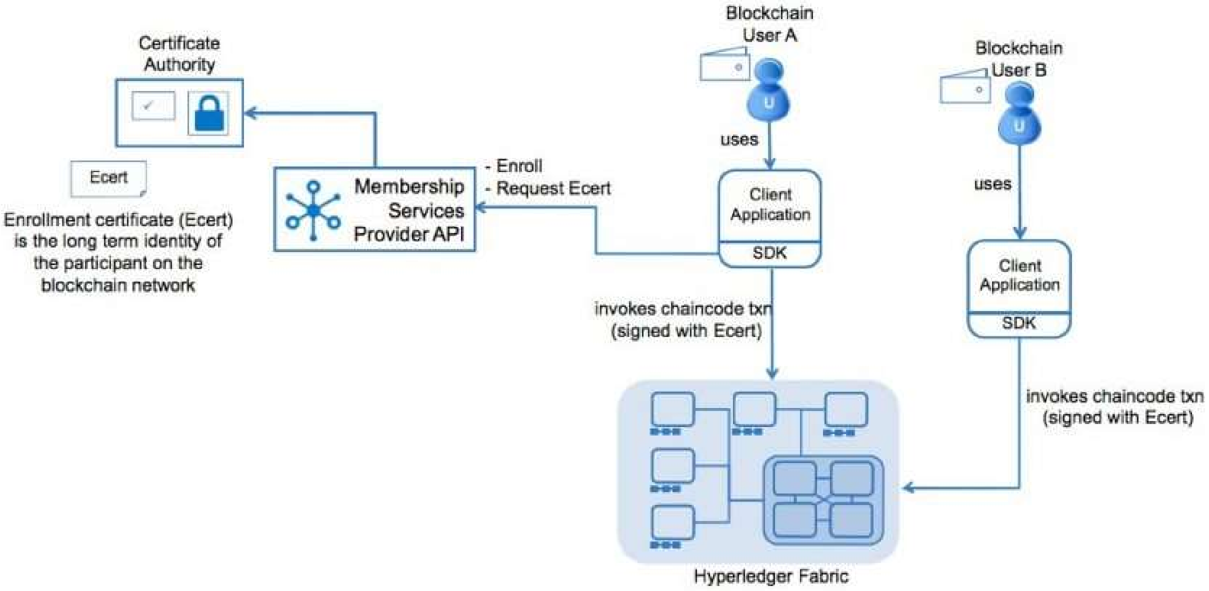
- HYPER LODGE IS A PRIVATE BLOCK CHAIN

- Hyperledger Fabric (for simplicity Fabric), has recently obtained massive popularity with hundreds of implementations around the world, since it is quite scalable, lenient against faults, and robust

- Nearly almost all the permissioned blockchain solutions can implement smart contracts, which are based on a programmable application logic that is being called each time a transaction is being proposed. In Fabric's case the smart contracts are realized by means of an arbitrary program that is authored in Go; the chaincode

- From the security perspective, we analyze Fabric into four interconnected components, in which possible attacks and leakage of private information can occur; namely: the consensus, the chaincode, the network and its privacy preservation mechanisms.

HYPERLEDGE FABRIC IN BLOCK CHAIN

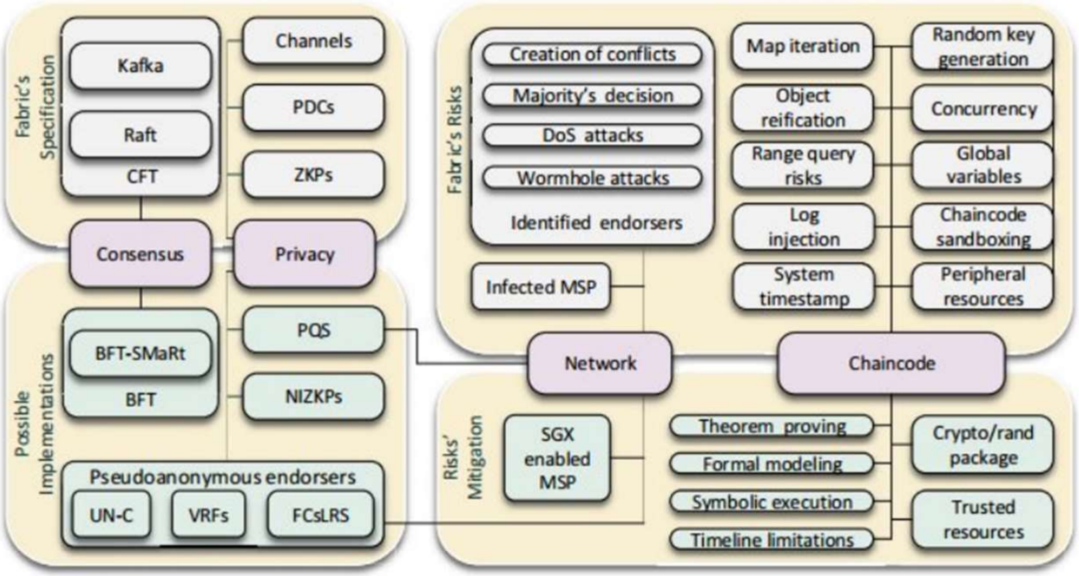


Certificate Authority
Ecert
Enrollment certificate (Ecert) is the long term identity of the participant on the blockchain network
Membership Services Provider API
- Enroll
- Request Ecert
Blockchain User A
U
uses
Client Application
SDK
invokes chaincode txn (signed with Ecert)
Blockchain User B
U
uses
Client Application
SDK
invokes chaincode txn (signed with Ecert)
Hyperledger Fabric

SOME Security Threats in Hyperledger Fabric

- Denial of service (DOS)
- MSP Compromise
- Consensus Manipulation
- Private Key Attacks
- Spoofing
- Algorithm Attacks
- Smart Contract Exploitation
- Ledger Manipulation
- Trojan Horse

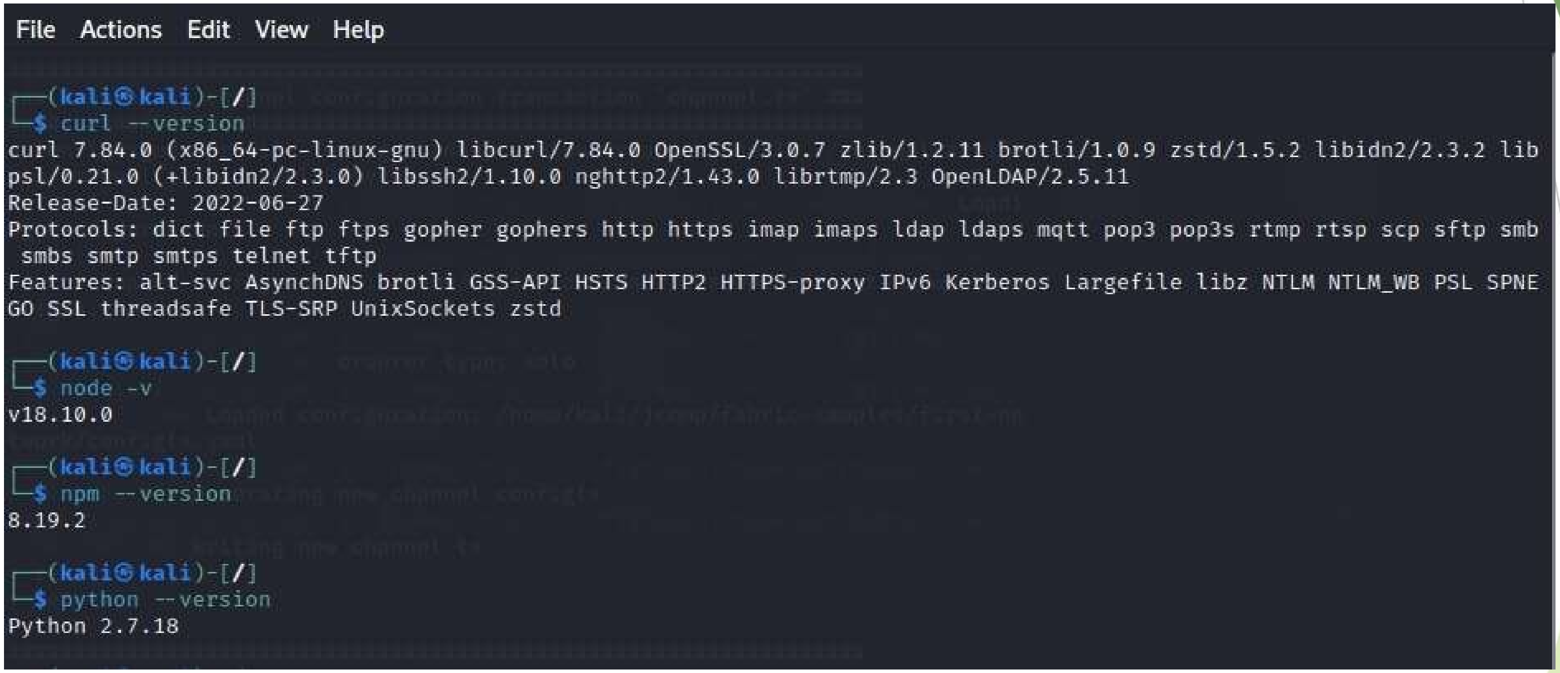
Structure of hyperledge fabric



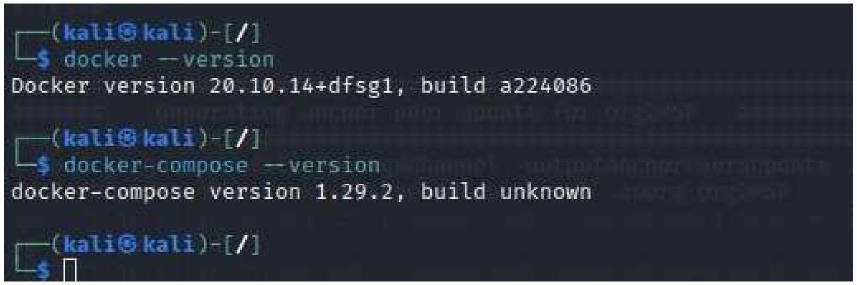
The chaincode is executed by a set of peers locally and before each transaction is appended into the ledger, an output of the chaincode's execution is taken into account in order to decide whether a transactions is valid or not and which data will be included to the ledger.

Implementation

- Installing required dependencies

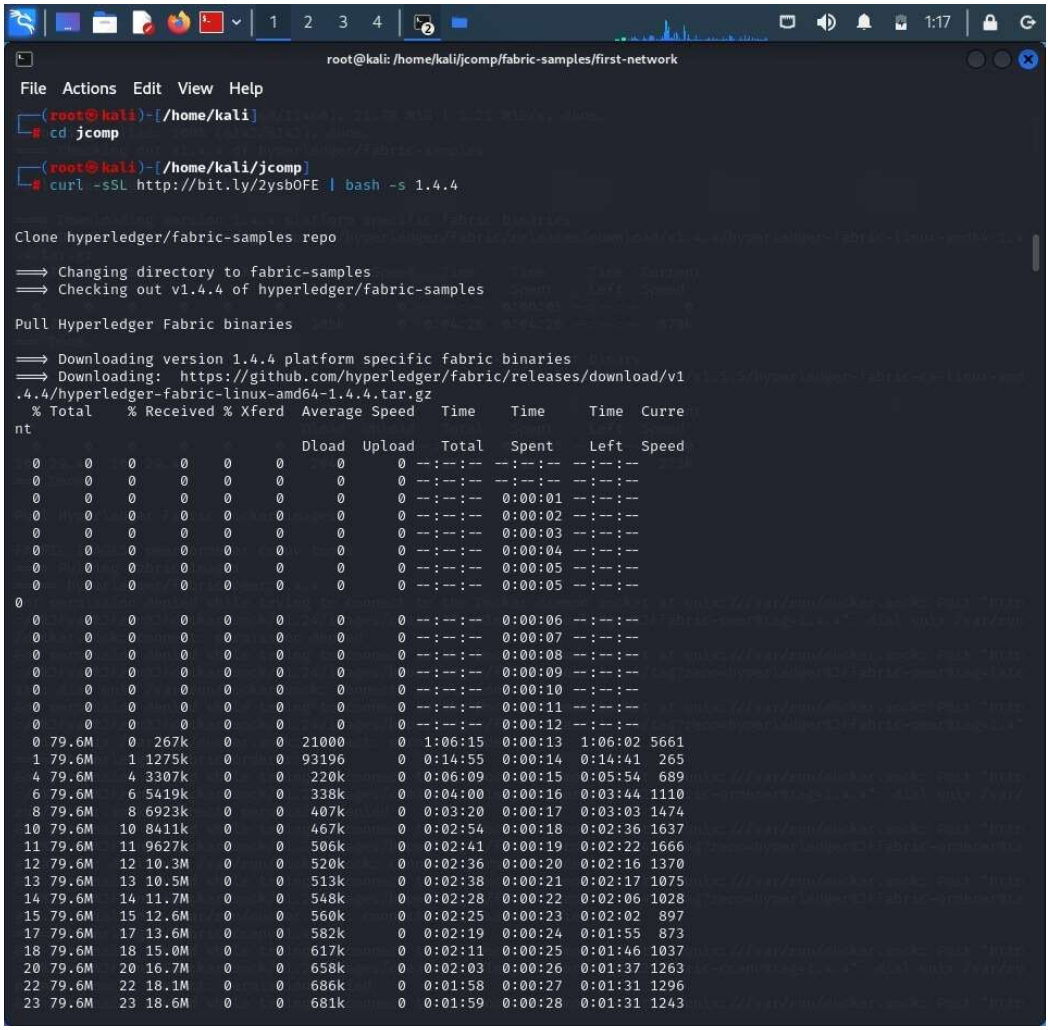


```
┌──(kali㉿kali)-[/]
└─$ curl --version
curl 7.84.0 (x86_64-pc-linux-gnu) libcurl/7.84.0 OpenSSL/3.0.7 zlib/1.2.11 brotli/1.0.9 zstd/1.5.2 libidn2/2.3.2 lib
psl/0.21.0 (+libidn2/2.3.0) libssh2/1.10.0 nghttp2/1.43.0 librtmp/2.3 OpenLDAP/2.5.11
Release-Date: 2022-06-27
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sftp smb
 smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNE
GO SSL threadsafe TLS-SRP UnixSockets zstd

┌──(kali㉿kali)-[/]
└─$ node -v
v18.10.0

┌──(kali㉿kali)-[/]
└─$ npm --version
8.19.2

┌──(kali㉿kali)-[/]
└─$ python --version
Python 2.7.18
```



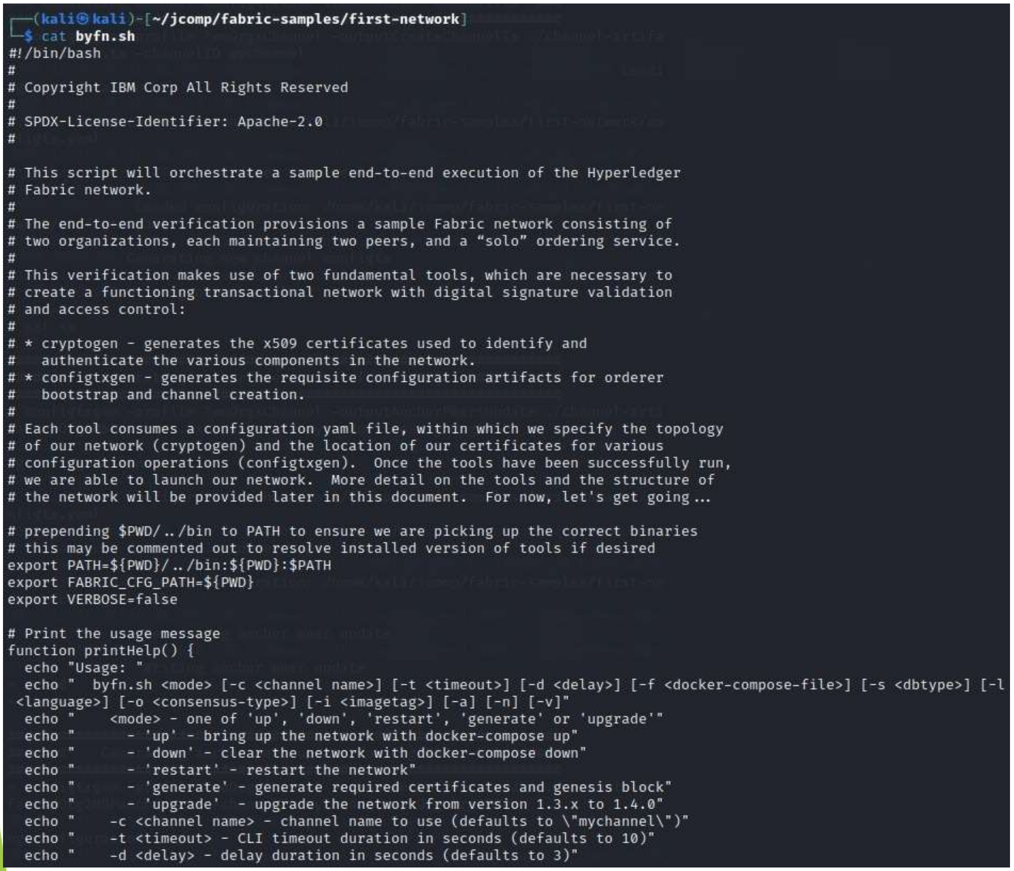
```
┌──(kali㉿kali)-[/]
└─$ docker --version
Docker version 20.10.14+dfsg1, build a224086

┌──(kali㉿kali)-[/]
└─$ docker-compose --version
docker-compose version 1.29.2, build unknown

┌──(kali㉿kali)-[/]
└─$ 
```

- Installing Hyperledger fabric and downloading docker images

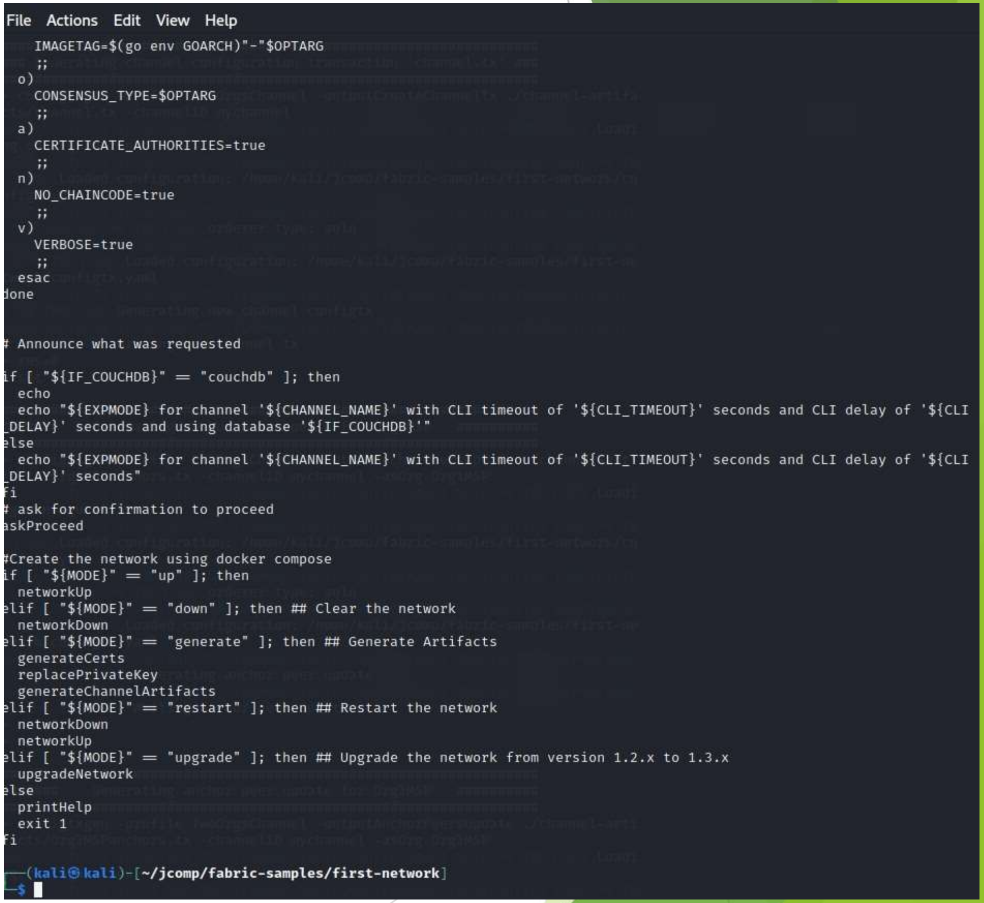


```
┌──(root㉿kali)-[/home/kali]
└─# cd jcomp

┌──(root㉿kali)-[/home/kali/jcomp]
└─# curl -sSL http://bit.ly/2ysbOFE | bash -s 1.4.4


Clone hyperledger/fabric-samples repo

===> Changing directory to fabric-samples
===> Checking out v1.4.4 of hyperledger/fabric-samples

Pull Hyperledger Fabric binaries

===> Downloading version 1.4.4 platform specific fabric binaries
===> Downloading:  https://github.com/hyperledger/fabric/releases/download/v1
.4.4/hyperledger-fabric-linux-amd64-1.4.4.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curre
nt
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:02 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:03 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:04 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:05 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:05 --:--:--
0
  0     0    0     0    0     0      0      0 --:--:--  0:00:06 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:07 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:08 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:09 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:10 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:11 --:--:--
  0     0    0     0    0     0      0      0 --:--:--  0:00:12 --:--:--
  0 79.6M    0  267k    0     0  21000      0  1:06:15  0:00:13  1:06:02 5661
  1 79.6M    1 1275k    0     0  93196      0  0:14:55  0:00:14  0:14:41  265
  4 79.6M    4 3307k    0     0   220k      0  0:06:09  0:00:15  0:05:54  689
  6 79.6M    6 5419k    0     0   338k      0  0:04:00  0:00:16  0:03:44 1110
  8 79.6M    8 6923k    0     0   407k      0  0:03:20  0:00:17  0:03:03 1474
 10 79.6M   10 8411k    0     0   467k      0  0:02:54  0:00:18  0:02:36 1637
 11 79.6M   11 9627k    0     0   506k      0  0:02:41  0:00:19  0:02:22 1666
 12 79.6M   12 10.3M    0     0   520k      0  0:02:36  0:00:20  0:02:16 1370
 13 79.6M   13 10.5M    0     0   513k      0  0:02:38  0:00:21  0:02:17 1075
 14 79.6M   14 11.7M    0     0   548k      0  0:02:28  0:00:22  0:02:06 1028
 15 79.6M   15 12.6M    0     0   560k      0  0:02:25  0:00:23  0:02:02  897
 17 79.6M   17 13.6M    0     0   582k      0  0:02:19  0:00:24  0:01:55  873
 18 79.6M   18 15.0M    0     0   617k      0  0:02:11  0:00:25  0:01:46 1037
 20 79.6M   20 16.7M    0     0   658k      0  0:02:03  0:00:26  0:01:37 1263
 22 79.6M   22 18.1M    0     0   686k      0  0:01:58  0:00:27  0:01:31 1296
 23 79.6M   23 18.6M    0     0   681k      0  0:01:59  0:00:28  0:01:31 1243
```

- Opening shell script that will create the new network

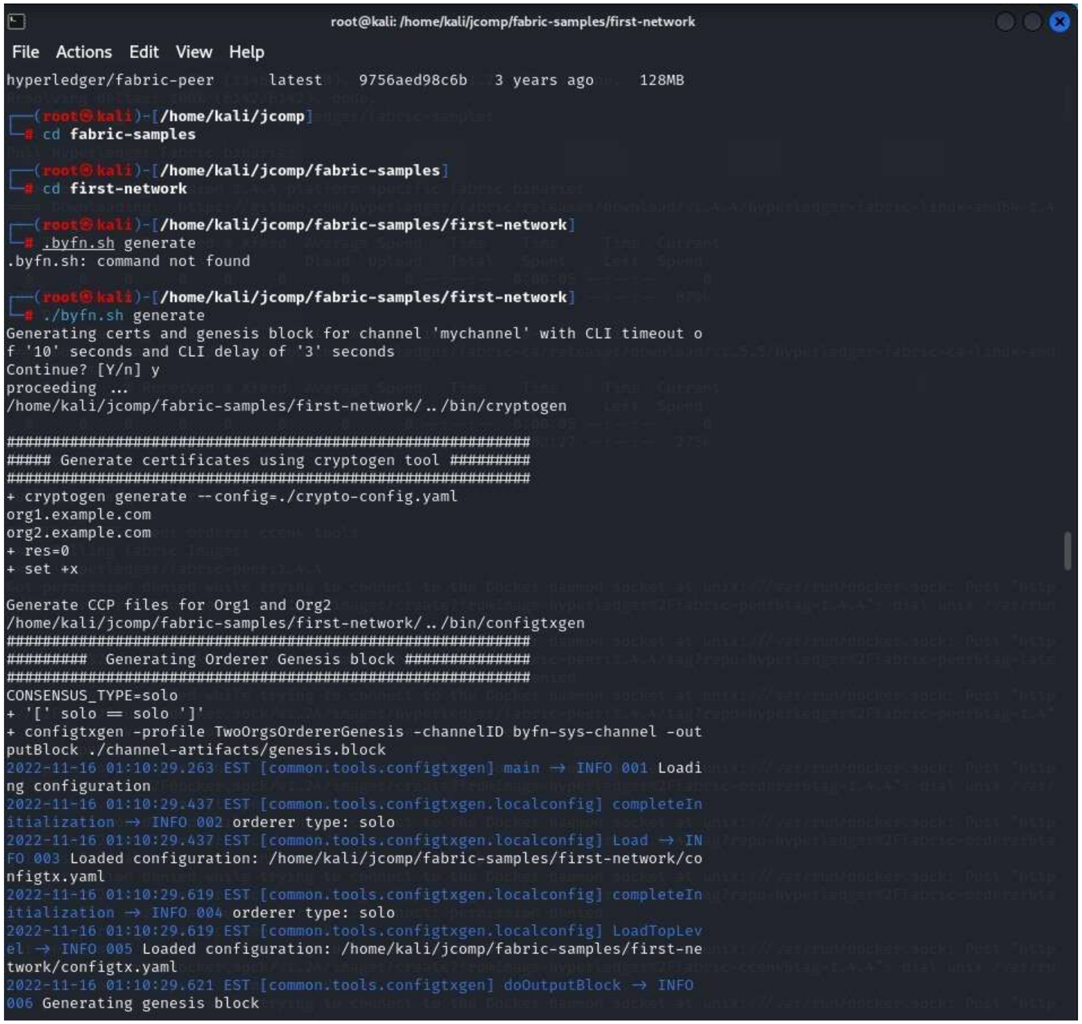


```
┌──(kali㉿kali)-[~/jcomp/fabric-samples/first-network]
└─$ cat byfn.sh
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#

# This script will orchestrate a sample end-to-end execution of the Hyperledger
# Fabric network.
#
# The end-to-end verification provisions a sample Fabric network consisting of
# two organizations, each maintaining two peers, and a “solo” ordering service.
#
# This verification makes use of two fundamental tools, which are necessary to
# create a functioning transactional network with digital signature validation
# and access control:
#
# * cryptogen - generates the x509 certificates used to identify and
#   authenticate the various components in the network.
# * configtxgen - generates the requisite configuration artifacts for orderer
#   bootstrap and channel creation.
#
# Each tool consumes a configuration yaml file, within which we specify the topology
# of our network (cryptogen) and the location of our certificates for various
# configuration operations (configtxgen).  Once the tools have been successfully run,
# we are able to launch our network.  More detail on the tools and the structure of
# the network will be provided later in this document.  For now, let's get going...

# prepending $PWD/../bin to PATH to ensure we are picking up the correct binaries
# this may be commented out to resolve installed version of tools if desired
export PATH=${PWD}/../bin:${PWD}:$PATH
export FABRIC_CFG_PATH=${PWD}
export VERBOSE=false

# Print the usage message
function printHelp() {
  echo "Usage: "
  echo "  byfn.sh <mode> [-c <channel name>] [-t <timeout>] [-d <delay>] [-f <docker-compose-file>] [-s <dbtype>] [-l
 <language>] [-o <consensus-type>] [-i <imagetag>] [-a] [-n] [-v]"
  echo "    <mode> - one of 'up', 'down', 'restart', 'generate' or 'upgrade'"
  echo "      - 'up' - bring up the network with docker-compose up"
  echo "      - 'down' - clear the network with docker-compose down"
  echo "      - 'restart' - restart the network"
  echo "      - 'generate' - generate required certificates and genesis block"
  echo "      - 'upgrade'  - upgrade the network from version 1.3.x to 1.4.0"
  echo "    -c <channel name> - channel name to use (defaults to \"mychannel\")"
  echo "    -t <timeout> - CLI timeout duration in seconds (defaults to 10)"
  echo "    -d <delay> - delay duration in seconds (defaults to 3)"
```



```
    IMAGETAG=$(go env GOARCH)"-"$OPTARG
    ;;
  o)
    CONSENSUS_TYPE=$OPTARG
    ;;
  a)
    CERTIFICATE_AUTHORITIES=true
    ;;
  n)
    NO_CHAINCODE=true
    ;;
  v)
    VERBOSE=true
    ;;
  esac
done

# Announce what was requested

if [ "${IF_COUCHDB}" == "couchdb" ]; then
  echo
  echo "${EXPMODE} for channel '${CHANNEL_NAME}' with CLI timeout of '${CLI_TIMEOUT}' seconds and CLI delay of '${CLI
_DELAY}' seconds and using database '${IF_COUCHDB}'"
else
  echo "${EXPMODE} for channel '${CHANNEL_NAME}' with CLI timeout of '${CLI_TIMEOUT}' seconds and CLI delay of '${CLI
_DELAY}' seconds"
fi
# ask for confirmation to proceed
askProceed

#Create the network using docker compose
if [ "${MODE}" == "up" ]; then
  networkUp
elif [ "${MODE}" == "down" ]; then ## Clear the network
  networkDown
elif [ "${MODE}" == "generate" ]; then ## Generate Artifacts
  generateCerts
  replacePrivateKey
  generateChannelArtifacts
elif [ "${MODE}" == "restart" ]; then ## Restart the network
  networkDown
  networkUp
elif [ "${MODE}" == "upgrade" ]; then ## Upgrade the network from version 1.2.x to 1.3.x
  upgradeNetwork
else
  printHelp
  exit 1
fi

┌──(kali㉿kali)-[~/jcomp/fabric-samples/first-network]
└─$ 
```

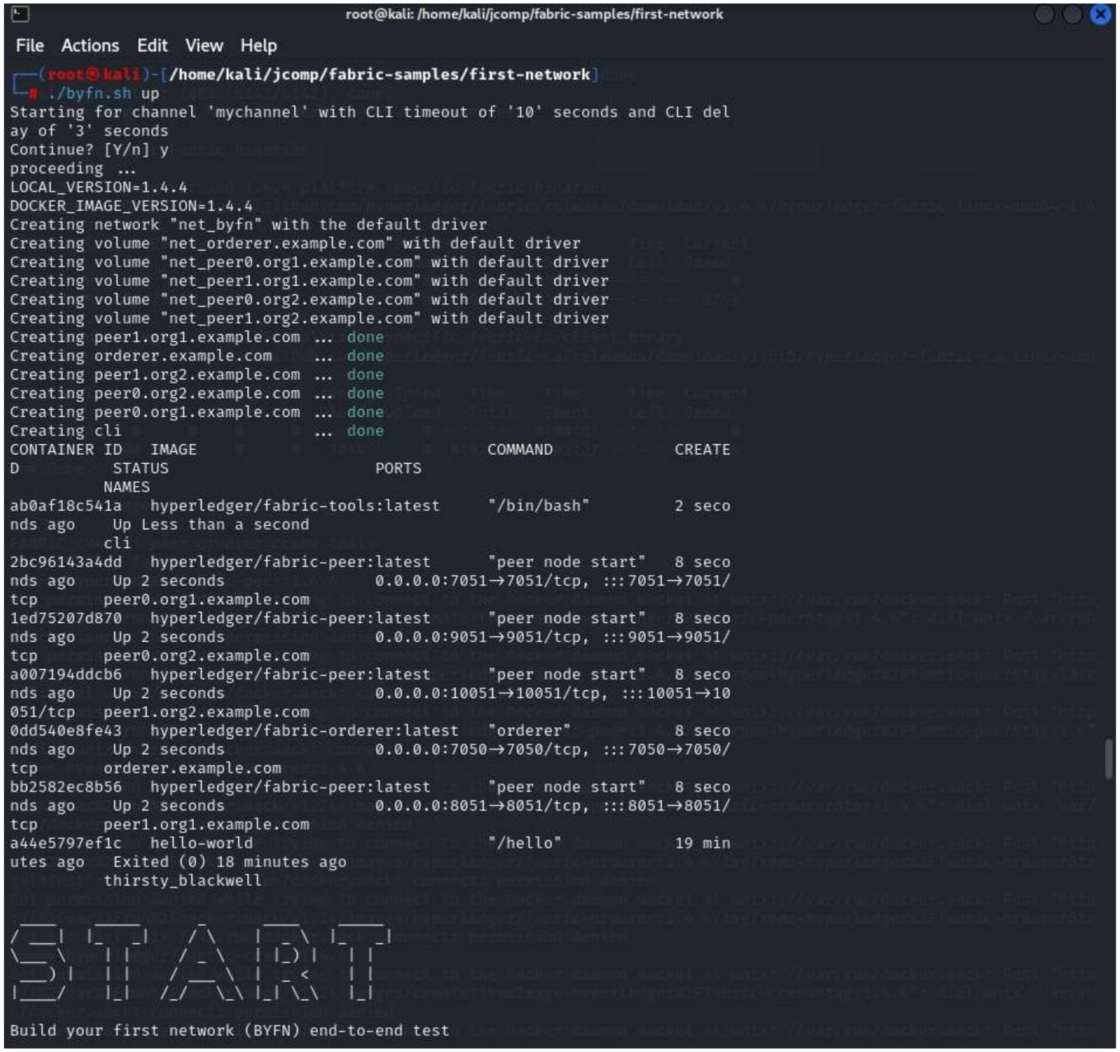
- Generating new network



root@kali: /home/kali/jcomp/fabric-samples/first-network
File Actions Edit View Help

```
hyperledger/fabric-peer      latest    9756aed98c6b   3 years ago    128MB

┌──(root㉿kali)-[/home/kali/jcomp]
└─# cd fabric-samples

┌──(root㉿kali)-[/home/kali/jcomp/fabric-samples]
└─# cd first-network

┌──(root㉿kali)-[/home/kali/jcomp/fabric-samples/first-network]
└─# .byfn.sh generate
.byfn.sh: command not found

┌──(root㉿kali)-[/home/kali/jcomp/fabric-samples/first-network]
└─# ./byfn.sh generate
Generating certs and genesis block for channel 'mychannel' with CLI timeout o
f '10' seconds and CLI delay of '3' seconds
Continue? [Y/n] y
proceeding ...
/home/kali/jcomp/fabric-samples/first-network/../bin/cryptogen

##########################################################
##### Generate certificates using cryptogen tool #########
##########################################################
+ cryptogen generate --config=./crypto-config.yaml
org1.example.com
org2.example.com
+ res=0
+ set +x

Generate CCP files for Org1 and Org2
/home/kali/jcomp/fabric-samples/first-network/../bin/configtxgen
##########################################################
#########  Generating Orderer Genesis block ##############
##########################################################
CONSENSUS_TYPE=solo
+ '[' solo = solo ']'
+ configtxgen -profile TwoOrgsOrdererGenesis -channelID byfn-sys-channel -out
putBlock ./channel-artifacts/genesis.block
2022-11-16 01:10:29.263 EST [common.tools.configtxgen] main -> INFO 001 Loadi
ng configuration
2022-11-16 01:10:29.437 EST [common.tools.configtxgen.localconfig] completeIn
itialization -> INFO 002 orderer type: solo
2022-11-16 01:10:29.437 EST [common.tools.configtxgen.localconfig] Load -> IN
FO 003 Loaded configuration: /home/kali/jcomp/fabric-samples/first-network/co
nfigtx.yaml
2022-11-16 01:10:29.619 EST [common.tools.configtxgen.localconfig] completeIn
itialization -> INFO 004 orderer type: solo
2022-11-16 01:10:29.619 EST [common.tools.configtxgen.localconfig] LoadTopLev
el -> INFO 005 Loaded configuration: /home/kali/jcomp/fabric-samples/first-ne
twork/configtx.yaml
2022-11-16 01:10:29.621 EST [common.tools.configtxgen] doOutputBlock -> INFO
006 Generating genesis block
```

- Turning on the network



```
┌──(root㉿kali)-[/home/kali/jcomp/fabric-samples/first-network]
└─# ./byfn.sh up
Starting for channel 'mychannel' with CLI timeout of '10' seconds and CLI del
ay of '3' seconds
Continue? [Y/n] y
proceeding ...
LOCAL_VERSION=1.4.4
DOCKER_IMAGE_VERSION=1.4.4
Creating network "net_byfn" with the default driver
Creating volume "net_orderer.example.com" with default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_peer1.org1.example.com" with default driver
Creating volume "net_peer0.org2.example.com" with default driver
Creating volume "net_peer1.org2.example.com" with default driver
Creating peer1.org1.example.com ... done
Creating orderer.example.com    ... done
Creating peer1.org2.example.com ... done
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
Creating cli                    ... done
CONTAINER ID   IMAGE                               COMMAND             CREATE
D         STATUS                        PORTS
          NAMES
ab0af18c541a   hyperledger/fabric-tools:latest     "/bin/bash"         2 seco
nds ago    Up Less than a second
          cli
2bc96143a4dd   hyperledger/fabric-peer:latest      "peer node start"   8 seco
nds ago    Up 2 seconds                 0.0.0.0:7051→7051/tcp, :::7051→7051/
tcp       peer0.org1.example.com
1ed75207d870   hyperledger/fabric-peer:latest      "peer node start"   8 seco
nds ago    Up 2 seconds                 0.0.0.0:9051→9051/tcp, :::9051→9051/
tcp       peer0.org2.example.com
a007194ddcb6   hyperledger/fabric-peer:latest      "peer node start"   8 seco
nds ago    Up 2 seconds                 0.0.0.0:10051→10051/tcp, :::10051→10
051/tcp   peer1.org2.example.com
0dd540e8fe43   hyperledger/fabric-orderer:latest   "orderer"           8 seco
nds ago    Up 2 seconds                 0.0.0.0:7050→7050/tcp, :::7050→7050/
tcp       orderer.example.com
bb2582ec8b56   hyperledger/fabric-peer:latest      "peer node start"   8 seco
nds ago    Up 2 seconds                 0.0.0.0:8051→8051/tcp, :::8051→8051/
tcp       peer1.org1.example.com
a44e5797ef1c   hello-world                         "/hello"            19 min
utes ago   Exited (0) 18 minutes ago
          thirsty_blackwell

 ____    _____      _      ____    _____
/ ___|  |_   _|    / \    |  _ \  |_   _|
\___ \    | |     / _ \   | |_) |   | |
 ___) |   | |    / ___ \  |  _ <    | |
|____/    |_|   /_/   \_\ |_| \_\   |_|

Build your first network (BYFN) end-to-end test
```

ATTACK ON SCALE

- Another conceivable assault vector includes suppositions about what an industrial-scale DPoS blockchain resembles
- It has not yet been seen by and by; be that as it may, on the off chance that it occurs, the suggestions merit considering
- EOS is probably going to scale such that huge server farms go about as BPs so as to give the degree of transmission capacity and speed the system requires.

GOVERNMENTAL

- GovernMental encounter experiences a comparable issue, though through subtler methods. The agreement was a Ponzi conspire. Clients would send Ether to the agreement with the guarantee of an expanded return and with the opportunity to win a "big stake."
- The agreement put away its clients' locations in a powerfully measured cluster and expected to repeat over the exhibits so as to clear them when a big stake was hit. In any case, it didn't constrain the size of the cluster.

WALLETSECURITY

- By and large, cryptographic forms of money store their incentive in a document store called a wallet, whereby every customer claims a lot of private-open keys to get to the wallet
- Clients regularly neglect to review their defensive PIN or secret phrase or lose the hard drive where the private key is found.
- Wallet burglary utilizes exemplary instruments, for example, phishing, which incorporates framework hacking, the establishment of surrey programming, and the erroneous utilization of wallets.
- A blockchain framework can without much of a stretch be abused through any powerlessness that may add to a cryptographic arrangement since clearly any programming bug or absence of secure private key can be the establishment of a significant security break

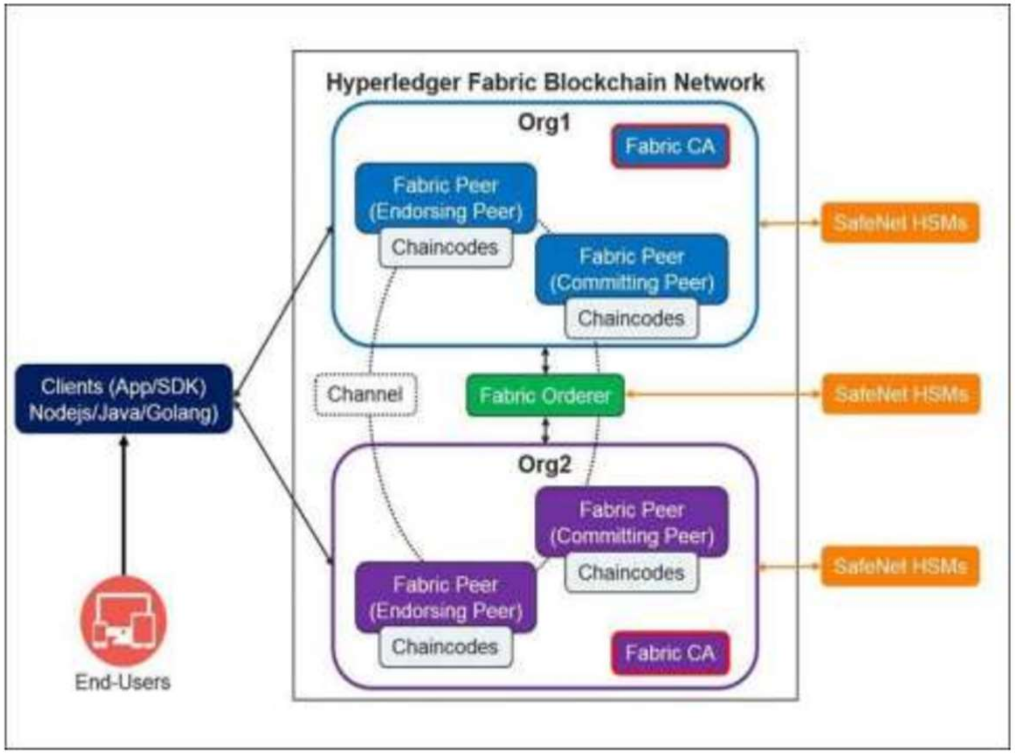
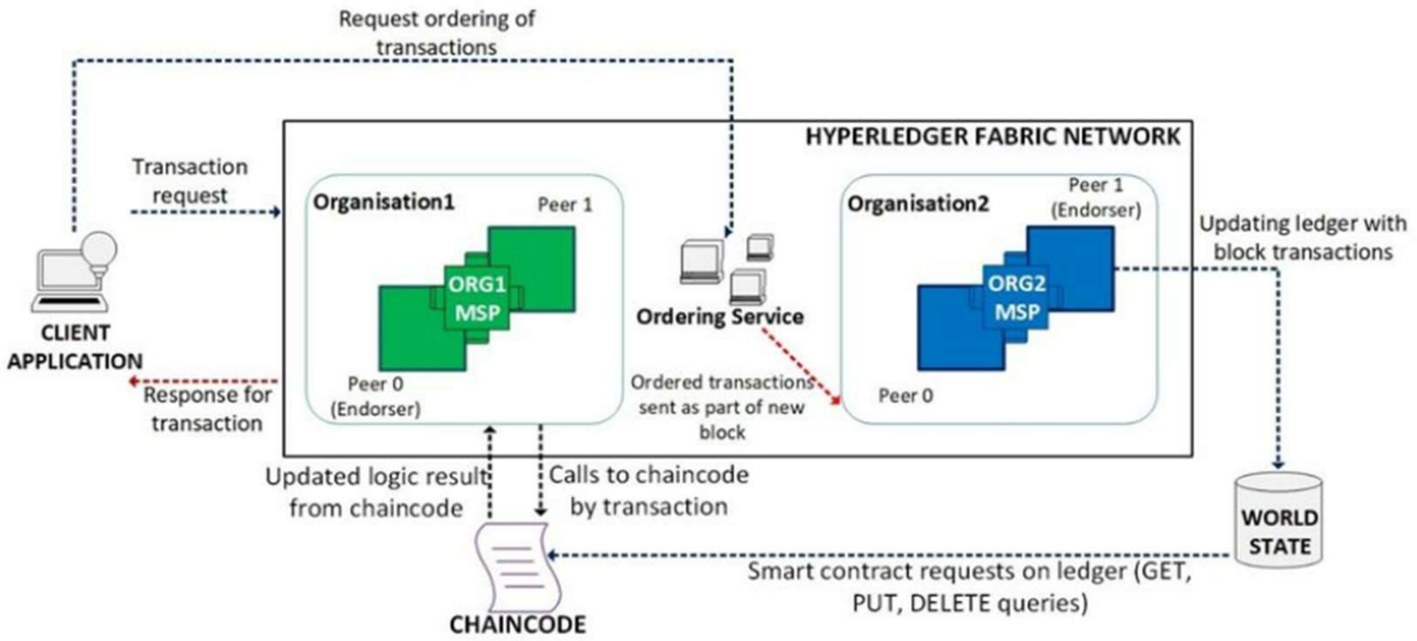

Hyperledger Fabric Blockchain Network
Org1
Fabric CA
Fabric Peer (Endorsing Peer)
Chaincodes
Fabric Peer (Committing Peer)
Chaincodes
SafeNet HSMs
Clients (App/SDK) Nodejs/Java/Golang)
Channel
Fabric Orderer
SafeNet HSMs
Org2
Fabric Peer (Committing Peer)
Chaincodes
SafeNet HSMs
Fabric Peer (Endorsing Peer)
Chaincodes
Fabric CA
End-Users

Fig. 8 Secure operation procedures on various End-to-End User

Wallet burglary utilizes exemplary instruments, for example, phishing, which incorporates framework hacking, the establishment of surrey programming, and the erroneous utilization of wallets.

Transactions within Hyperledger fabric network


Request ordering of transactions
HYPERLEDGER FABRIC NETWORK
Transaction request
Organisation1
Peer 1
ORG1 MSP
Peer 0 (Endorser)
Ordering Service
Ordered transactions sent as part of new block
Organisation2
Peer 1 (Endorser)
ORG2 MSP
Peer 0
Updating ledger with block transactions
CLIENT APPLICATION
Response for transaction
Updated logic result from chaincode
Calls to chaincode by transaction
WORLD STATE
Smart contract requests on ledger (GET, PUT, DELETE queries)
CHAINCODE

- National Bank of Cambodia - leveraging on Hyperledger Iroha to solve country’s banking challenges WHICH include creation of a modern digital payment system that is fast and secure
- Walmart – creating transparency within their supply chain through use of Hyperledger Fabric, this project was undertaken through partnership with IBM
- . British Columbia – British Columbia and Ontario governments looked at ways to help minimise government red tape by using software stack to empower businesses to establish trusted and enduring relationship. The blockchain framework used for this project is Hyperledger Indy and system was announced in January 2019

THANK YOU