KC Tam

May 13, 2020 · 21 min read · Listen

Ways to Generate Crypto Materials in erledger Fabric: Cryptogen and CA Server

ew

he most asked questions on Hyperledger Fabric is about identity. A ioned blockchain requires that an entity, be it a client, an trator or a network component, must be first identified and ioned before accessing a consortium network. Hyperledger Fabric eral concepts like organization, membership service providers, ertificate, which makes the understanding of the whole picture a allenging for new learners. Here in this article again I try to elaborate ncepts through analyzing Test Network, a network example in fabric l, and the two different ways to generate the crypto material for ıtities. I hope to give you another perspective on this topic.

zation and Membership Service Provider

dger Fabric is always chosen to build a consortium network to a business problem. The participants are usually business entities in e. An example is a trade finance platform, where participants are he banks, plus some governance or regulatory bodies. Hyperledger ıses **organizations** to represent these participating entities. In the ıance platform, each bank is an organization. A governance body is other organization. This is how we understand organization from a s perspective.

rledger Fabric, each organization is characterized by its Membership Provider (MSP). On one side, MSP creates credential material to all within an organization. On the other side, MSP represents the ation to participate in a consortium network. Technically the ium network is formed by the MSP of all organizations. They trust ther when the MSPs are functioning well.

mplemented in Public Key Infrastructure (PKI). In a typical PKI ment each entity is holding a **private key** and a **digital certificate ate)** containing the public key and relevant information about this his certificate is signed and issued by a **Certificate Authority (CA)** omes the identity of this entity.

n entity signs a message with its private key, it produces a signature message. Everyone who obtain this message, signature and the certificate, can

hat only the private key owner can create this signature (through ic key from certificate)

v the identity of the private key owner (through information rded in the certificate)

: that the signer is the one claimed in the certificate (through ying CA's signature on the certificate, with the assumption that CA's ficate is trusted).

ning and verification process happens everywhere in a consortium t. Each organization knows the MSP of other organizations in a ium network, and trust of this consortium network is built up from a vho" perspective. As an example, here is a part of a transaction ; the endorsement, and we can see those information mentioned

A capture of a transaction showing identity of an entity (endorser) and the signature.

ransaction two endorsements are included. Each endorsement
s

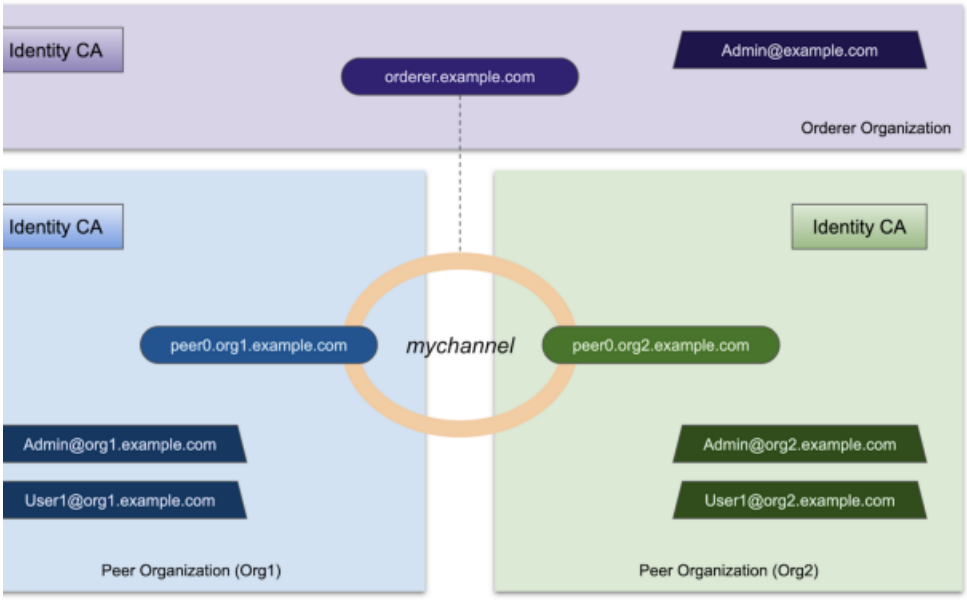
rser: the certificate (identity) of endorser in base64

ature: produced by that endorser against the message in *proposal nse payload* (collapsed).

e in the consortium can verify that it is the endorser in the te signing this proposal response, and therefore this endorsement is d.

etwork

ays too abstract if just reading the material without examples. Here *Test Network* from fabric samples as an example. Similar to First x in previous releases, Test Network comes with a well-designed fabric v2.0. It is a three-organization design, with one orderer ation and two peer organizations.



Test Network: Organization and entities inside each organization

ral the structure of an organization includes

, or we use the term Identity CA (or simply CA) representing it

vork components, either orderers or peers

vork users, who represent the actual human or client application
ıg on the network

network components and *network users* separately if the discussion is
to each of them. If something applies to both network components
work users, we will use the term ***entities*** of an organization.

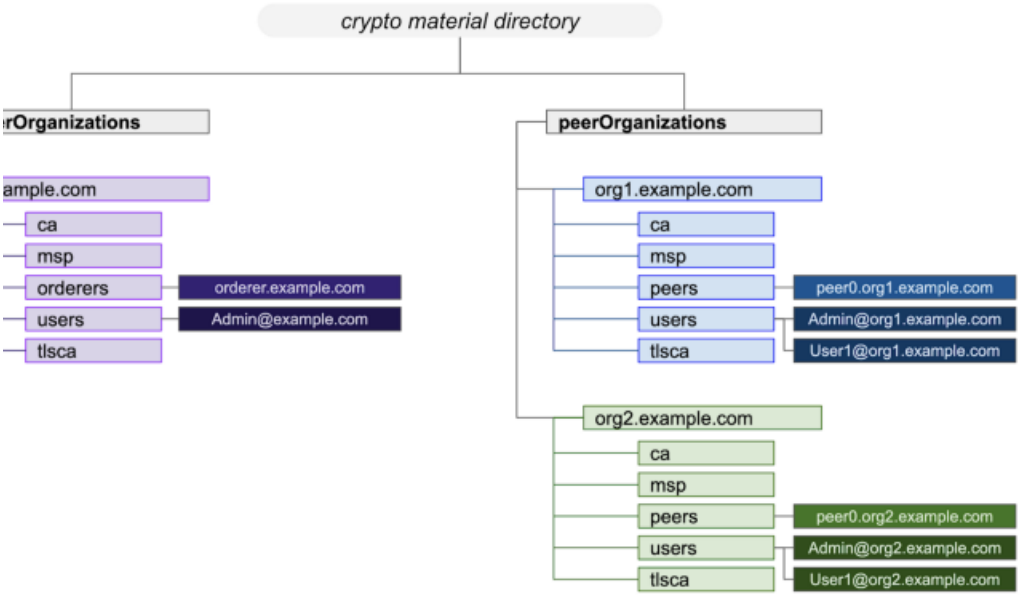
sponsible for handling the identity of all entities within an
ation. Each entity obtains a credential (crypto material) from CA. The
ial includes a private key and a digital certificate representing this
'he security is maintained as long as the entity is keeping the private
.

ıe main purpose of CA is on identity area (prove an entity is what it
o be), we will encounter another set of crypto material for another

:. In Hyperledger Fabric communication between components are ured by TLS, for example, client application to peers, cluster nication between orderers, etc. Although this article is more focused tity, we will also take a look on TLS, but you should always see them eparate systems for different purposes.

about Directory Structure of Crypto Material

se who have gained some experience in First Network in the previous or Test Network in v2.0, you may have noticed that the crypto l is always stored in a fixed directory structure.



examine the directory in detail, but I wish to point out that there is ust to follow such a data structure. If we trace back a bit history, at nning we are using `cryptogen` (more detail in next session) to create to material, and `docker-compose` to bring up network components. he docker-compose configuration files directories are mapped to twork component. For example, crypto material for `rg1.example.com` is correctly installed in this peer through docker- e files.

```
r0.org1.example.com:
ontainer_name: peer0.org1.example.com
mage: hyperledger/fabric-peer:$IMAGE_TAG
nvironment:-
olumes:
  - /var/run/:/host/var/run/
  - ../organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/fabric/msp
  - ../organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls:/etc/hyperledger/fabric/tls
  - peer0.org1.example.com:/var/hyperledger/production
orking_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
ommand: peer node start
orts:
 - 7051:7051
etworks:
 - test
```

o material of peer0.org1 is correctly mapped to peer0.org1 containers in docker-compose files

ecent and systematic way as it can be coded into configuration. eless, you also can mount the crypto material one-by-one and ly, if it is desired.

ill rely on docker-compose files for bringing up network ients, *we will follow this directory structure*. It is automatically ed with **`cryptogen`**, but some more copy-n-paste is needed when we g CA Server.

ese, we can start looking into the first way to generate crypto l: **`cryptogen`**.

ting Crypto Material using Cryptogen

dger Fabric provides a tool that crypto material can be generated nimum configuration. The tool is **`bin/cryptogen`**. Working with a ration file, the crypto material of Test Network is generated and the kept as the directory structure shown above. With that, we can bring onsortium network with docker compose files.

ry Structure

use `./network.sh` script to bring up Test Network and make some tions on the directory structure.

```
st-network
work.sh up
```

pt design the material is kept in `organizations/`. Here we take Org1 as
ıple for exploration.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com

2 ubuntu ubuntu 4096 May 10 03:03 ca
1 ubuntu ubuntu 2959 May 10 03:03 connection-org1.json
1 ubuntu ubuntu 2645 May 10 03:03 connection-org1.yaml
5 ubuntu ubuntu 4096 May 10 03:03 msp
3 ubuntu ubuntu 4096 May 10 03:03 peers
2 ubuntu ubuntu 4096 May 10 03:03 tlsca
4 ubuntu ubuntu 4096 May 10 03:03 users
```

e a quick summary for all these components

a directory holding crypto material for CA of Org1

s/ : a directory holding a list of peers under Org1, and each peer has
wn crypto material, of both identity and TLS

s/ : a directory holding a list of network users under Org1, and each
has its own crypto material, of both identity and TLS

a/ : a directory holding the TLS CA

: it is the material joining a consortium network.

ection-org1.yaml and connection-org1.json : these are the connection
ile files in different formats. They will be used in client applications.

k it further down and observe some relevant items.

ficate

take a look at the CA of Org1. It is this CA issuing identity
ate) to all entities within Org1.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
/

 1 ubuntu ubuntu 863 May 10 03:03 ca.org1.example.com-cert.pem
 1 ubuntu ubuntu 241 May 10 03:03 priv_sk
```

e directory we see a private secret key and a certificate. We will the subject and issuer of the certificate.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
x509 -in ca/ca.org1.example.com-cert.pem -noout -subject -issuer
= US, ST = California, L = San Francisco, O = org1.example.com, CN = ca.org1.example.com
 US, ST = California, L = San Francisco, O = org1.example.com, CN = ca.org1.example.com
```

lf-signed certificate (subject == issuer). It is a typical design for non-ion environments, as we allow a new self-signed certificate as the CA identity for the whole organization. This is what `cryptogen` can , and this is also its limitation (see discussion later).

: *Components (peer)*

is a peer organization, the network component is “peer”. In the we see a list of peers defined per the crypto configuration file (will be ed later). In the Test Network we have only one peer.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
ers/

 4 ubuntu ubuntu 4096 May 10 03:03 peer0.org1.example.com
```

click to this directory, there are two directories. **`msp/`** is for identity, **`/`** is for TLS material. This directory structure will be seen in every

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
r0.org1.example.com$ ls -l

 7 ubuntu ubuntu 4096 May 10 03:03 msp
 2 ubuntu ubuntu 4096 May 10 03:03 tls
```

s on the identity part. We will revisit the TLS material when we cover
r.

sp/ the directory structure looks like this. We will be very
ized with this directory structure later, as it is the structure for all
both network components and network users.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
r0.org1.example.com$ ls -l msp/

2 ubuntu ubuntu 4096 May 10 03:03 admincerts
2 ubuntu ubuntu 4096 May 10 03:03 cacerts
1 ubuntu ubuntu  488 May 10 03:03 config.yaml
2 ubuntu ubuntu 4096 May 10 03:03 keystore
2 ubuntu ubuntu 4096 May 10 03:03 signcerts
2 ubuntu ubuntu 4096 May 10 03:03 tlscacerts
```

ıtity, the private secret key of this peer is stored in **`msp/keystore/`**,
e certificate is kept in **`msp/signcerts/`**.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
r0.org1.example.com$ ls -l msp/keystore/

1 ubuntu ubuntu 241 May 10 03:03 priv_sk
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
r0.org1.example.com$ ls -l msp/signcerts/

1 ubuntu ubuntu 810 May 10 03:03 peer0.org1.example.com-cert.pem
```

a look at the certificate of this peer. The issuer is the CA of Org1
ıed above, and the subject is **`CN=peer0.org1.example.com`** and **`OU=peer`**.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
r0.org1.example.com$ openssl x509 -in msp/signcerts/peer0.org1.example.com-cert.pem -noout
issuer
= US, ST = California, L = San Francisco, OU = peer, CN = peer0.org1.example.com
 US, ST = California, L = San Francisco, O = org1.example.com, CN = ca.org1.example.com
```

Users

users are kept in `users/`. From the directory we see a list of users
in the crypto configuration file (will be examined later). In Test

x we have one **Admin**, and one user named **User1**.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
ers/

 4 ubuntu ubuntu 4096 May 10 03:03 Admin@org1.example.com
 4 ubuntu ubuntu 4096 May 10 03:03 User1@org1.example.com
```

e Admin as an example. Again there are two directories. **`msp/`** is for , and **`tls/`** is for TLS material.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
in@org1.example.com$ ls -l

 7 ubuntu ubuntu 4096 May 10 03:03 msp
 2 ubuntu ubuntu 4096 May 10 03:03 tls
```

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
in@org1.example.com$ ls -l msp/

 2 ubuntu ubuntu 4096 May 10 03:03 admincerts
 2 ubuntu ubuntu 4096 May 10 03:03 cacerts
 1 ubuntu ubuntu  488 May 10 03:03 config.yaml
 2 ubuntu ubuntu 4096 May 10 03:03 keystore
 2 ubuntu ubuntu 4096 May 10 03:03 signcerts
 2 ubuntu ubuntu 4096 May 10 03:03 tlscacerts
```

the private key and certificate for Admin.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
in@org1.example.com$ ls -l msp/keystore/

 1 ubuntu ubuntu 241 May 10 03:03 priv_sk
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
in@org1.example.com$ ls -l msp/signcerts/

 1 ubuntu ubuntu 810 May 10 03:03 Admin@org1.example.com-cert.pem
```

we take a look at the certificate of Admin, both the subject and issuer. hat the subject contains **`OU=admin`** , and the certificate is issued by the rg1.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
in@org1.example.com$ openssl x509 -in msp/signcerts/Admin@org1.example.com-cert.pem -noout
issuer
= US, ST = California, L = San Francisco, OU = admin, CN = Admin@org1.example.com
 US, ST = California, L = San Francisco, O = org1.example.com, CN = ca.org1.example.com
```

rectory structure for User1. There is a difference in the certificate We see **`OU=client`** for User1. Obviously it is the difference of roles in o network users. Admin is an admin, while User1 is a client user. If tions only allow admin role to perform, User1 is denied.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
r1@org1.example.com$ openssl x509 -in msp/signcerts/User1@org1.example.com-cert.pem -noout
issuer
= US, ST = California, L = San Francisco, OU = client, CN = User1@org1.example.com
 US, ST = California, L = San Francisco, O = org1.example.com, CN = ca.org1.example.com
```

TLS is not relevant to the identity system, the crypto material for TLS enerated during the cryptogen process. Here we just make a t and locate those materials inside the directory structure.

A is also created for Org1. The TLS CA Certificate is stored in **tlsca/**. pto material for all entities in Org1 is generated from this TLS CA.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
sca/

 1 ubuntu ubuntu 241 May 10 03:03 priv_sk
 1 ubuntu ubuntu 875 May 10 03:03 tlsca.org1.example.com-cert.pem
```

his is a self-signed certificate. Therefore in `cryptogen` the Identity TLS CA are different systems, and do not have any relationship to ıer.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
x509 -in tlsca/tlsca.org1.example.com-cert.pem -noout -subject -issuer
= US, ST = California, L = San Francisco, O = org1.example.com, CN = tlsca.org1.example.com
 US, ST = California, L = San Francisco, O = org1.example.com, CN = tlsca.org1.example.com
```

; communication, network components play a TLS server role.
re each network component is installed with a TLS server key and
ertificate. As in Org1 we only have one peer, we can inspect the
l inside **`peers/peer0.org1.example.com/tls/`**.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
r0.org1.example.com$ ls -l tls/

 1 ubuntu ubuntu  875 May 10 03:03 ca.crt
 1 ubuntu ubuntu 1025 May 10 03:03 server.crt
 1 ubuntu ubuntu  241 May 10 03:03 server.key
```

in we see the TLS server certificate **`server.crt`** is issued by TLS CA.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
r0.org1.example.com$ openssl x509 -in tls/server.crt -noout -subject -issuer
= US, ST = California, L = San Francisco, CN = peer0.org1.example.com
 US, ST = California, L = San Francisco, O = org1.example.com, CN = tlsca.org1.example.com
```

x user plays a TLS client role. Therefore each network user is
l with a TLS client key and client certificate. We take Admin as an
e here.

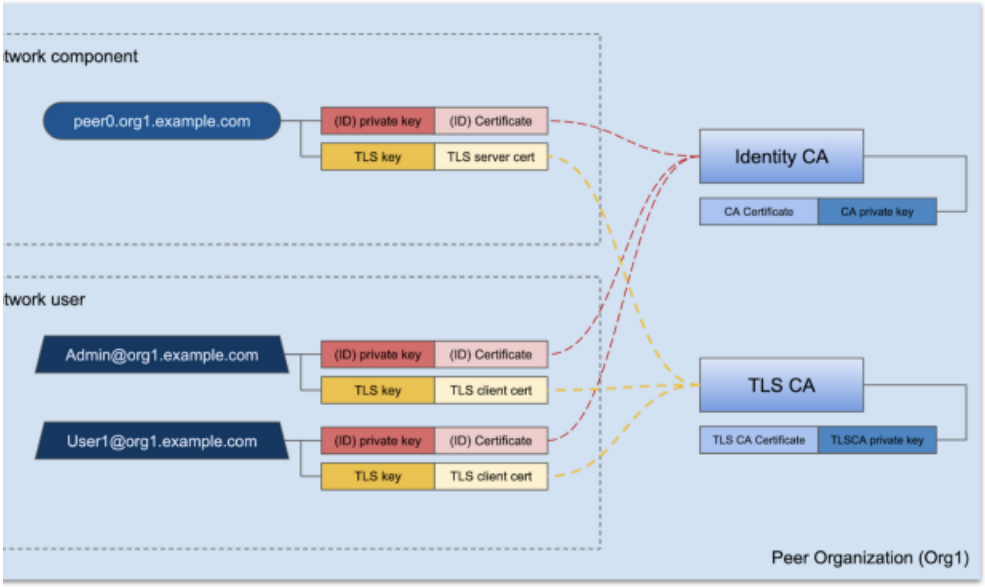
```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
in@org1.example.com$ ls -l tls/

 1 ubuntu ubuntu 875 May 10 03:03 ca.crt
 1 ubuntu ubuntu 834 May 10 03:03 client.crt
 1 ubuntu ubuntu 241 May 10 03:03 client.key
```

client certificate **`client.crt`** is also issued by TLS CA.

```
172-31-31-50:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com
in@org1.example.com$ openssl x509 -in tls/client.crt -noout -subject -issuer
= US, ST = California, L = San Francisco, CN = Admin@org1.example.com
 US, ST = California, L = San Francisco, O = org1.example.com, CN = tlsca.org1.example.com
```

diagram showing the relationship of crypto material generated with
en . Note that the Identity CA for each organization is just a CA's
key and a CA Certificate. There is no software currently running as a
er. You will not see a container running in any CA Servers generating

is certificate issuance process is done by `cryptogen` software itself. eless, one can easily bring up a CA Server with CA's private key and ificate when identity for more entities is needed.



Crypto material generated using `cryptogen` tool in Test Network

ıration File

e seen the result of `cryptogen`. Now we take a look at the ration files, which tell `cryptogen` how to generate the crypto material ıbove.

figuration for Test Network is kept inside **`organizations/cryptogen/`**. hese files we can specify the organization structure of the network ganization and orderer organization), and detail of entities within an ation. These configuration files are self-explanatory. For sake of eness we put them here.

erer organization. The result is one orderer with name **`.example.com`**. One **Admin** user is generated by default.

```
OrdererOrgs:
  - Name: Orderer
    Domain: example.com
    EnableNodeOUs: true
    Specs:
      - Hostname: orderer
        SANS:
          - localhost
```

configuration for Orderer Organization in Test Network

r organization (here we see Org1). The result is one peer (*Count 1* in e) and the name is by default **`peer0.org1.example.com`**. One **Admin** generated by default. The *Count 1* in Users means one additional user ated as well, with a name **User1** by default.

```
PeerOrgs:
  - Name: Org1
    Domain: org1.example.com
    EnableNodeOUs: true
    Template:
      Count: 1
      SANS:
        - localhost
    Users:
      Count: 1
```

configuration for Org1, a PeerOrganization in Test Network

tches all the entities we see above in Org1.

cript **`network.sh`**, the crypto material is generated in this command.

```
ogen generate --config=<configuration_file> --output=
ut_directory>
```

KC Tam
2.3K Followers

nmand is executed for all the three organizations: Org1, Org2 and Organization.

see how `cryptogen` works: with this command and configuration can build the directory structure of crypto material shown above.

ion

en by far is the simplest way for us to generate crypto material for network designs. No matter how many orderers we are to deploy (one , five for Raft, etc) in orderer organization, how many peers needed peer organization, or how many client users needed in each ation, we can specify the requirement in the configuration files and e the material with a simple command.

lso where the limitation lies. From Hyperledger Fabric ntation, we learn that `cryptogen` is good for generating crypto l for testing purposes, and would normally not be used in production on. There are several considerations.

hat the CA certificates are all generated as self-signed certificates. In we may ask for different settings. For example, we cannot use an CA as the MSP for our organization, or we cannot chained ates.

s predefined with organization type: orderer organizations and peer ations are defined separately. If for example we need orderers owned organizations, we have to use some workaround (refer to this article equirement).

not the least, material is generated once based on configuration file, eal life we may need more peers or more clients after deployment. h it is still doable, as we can generate the new entities with CA's key, it is not as simple as running `cryptogen` on a configuration file.

re just creating for testing, the above should not be your concern, and `en` is good enough. Otherwise, you can take another approach: bring server and generate the crypto material for entities with this CA ng to your desired network design. This is the second way to generate naterial for a consortium network.

ıting Crypto Material using CA Server

some limitations in using `cryptogen`. To gain more flexibility and more practical, a better way is to bring up a CA Server, and we can e crypto material for entities according to our network design. This more steps, but provides a standard way to either generate crypto l that `cryptogen` cannot generate, or for future use if more network ıents and users are needed.

ew

rver is a software to generate crypto material on request. Inside a CA here is a CA, represented by a private secret key and a CA Certificate. tificates signed and issued by this CA is verifiable with the CA ate (or CA's public key). CA's private secret key and CA Certificate is nce of a CA, while a CA Server is just a software tool to perform this

ally speaking, once one gets hold the CA's private secret key and CA ate, one can issue certificates. It can be done on a CA Server, or it can by common tools such as openssl. Therefore, always make a ce between a CA (representing the ownership of that private key) A Server (where certificate is signed and issued).

re many CA Server products in the market. Hyperledger Fabric vith one, **Fabric-CA Server**. This whole process begins with bringing oric-CA Server, running as a docker container. We either use an

CA for this CA Server, or generate a self-signed certificate as root
he Test Network example, it is a self-signed certificate, and we then
e all crypto material from it.

act with Fabric-CA Server, we need another tool on our side. It can be
Fabric-CA Client or Fabric SDKs. Fabric-CA Client is used in Test
x script. Note that Fabric-CA Client is running on our localhost, and it
s with the Fabric-CA Server which is a running container. You can
a comprehensive explanation in fabric documentation (link).

ry Structure and Content

tioned before, we need the same directory structure in order to
he docker-compose configuration. Therefore the result of using CA
be the same as the result after using `cryptogen`. This involves some
anging and file moving. Instead of repeating everything about the
y structure, here we highlight some difference on the content and
.

use **`./network.sh`** script to bring up the minimum setup and observe
lting directory structure and content. Note the option **`-a`** is required
up crypto material using CA instead of `cryptogen`.

```
st-network
work.sh up -ca
```

jority of steps are in fact executed in the script **`fabric-
terEnroll.sh`**.

three CAs, one for each organization, up and running. As a
ison, no CA containers are running when we bring up the network
`yptogen`.

```
172-31-28-82:~/fabric-samples/test-network$ docker ps --filter="name=ca"
ID        IMAGE                          COMMAND                  CREATED              STA
      PORTS                               NAMES
be        hyperledger/fabric-ca:latest   "sh -c 'fabric-ca-se…"   About a minute ago   Up
nute   7054/tcp, 0.0.0.0:8054->8054/tcp   ca_org2
78        hyperledger/fabric-ca:latest   "sh -c 'fabric-ca-se…"   About a minute ago   Up
nute   0.0.0.0:7054->7054/tcp              ca_org1
91        hyperledger/fabric-ca:latest   "sh -c 'fabric-ca-se…"   About a minute ago   Up
nute   7054/tcp, 0.0.0.0:9054->9054/tcp   ca_orderer
```

e CA Certificate is here, which is a public information, but there is no secret key (scroll up to the session on `cryptogen` and see a ison). It is because the private secret key is in the CA Server. This is a alistic way as it is the CA Server holding that secret, and it should not able and visible outside the CA Server. We will see more detail about Certificate when we dive into the Fabric-CA Server

icture for entities (peer and users) are similar. Here we just bring out ificate of the peer and both users. They are all issued by CA of Org1, per identifiers in OU.

```
172-31-28-82:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.co
 x509 -in peers/peer0.org1.example.com/msp/signcerts/cert.pem -noout -subject -issuer
= US, ST = North Carolina, O = Hyperledger, OU = peer, CN = peer0
 US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
172-31-28-82:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.co
 x509 -in users/Admin@org1.example.com/msp/signcerts/cert.pem -noout -subject -issuer
= US, ST = North Carolina, O = Hyperledger, OU = admin, CN = org1admin
 US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
172-31-28-82:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.co
 x509 -in users/User1@org1.example.com/msp/signcerts/cert.pem -noout -subject -issuer
= US, ST = North Carolina, O = Hyperledger, OU = client, CN = user1
 US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
```

take a look at the TLS. First the TLS CA. Interesting enough, we see e certificate as CA.

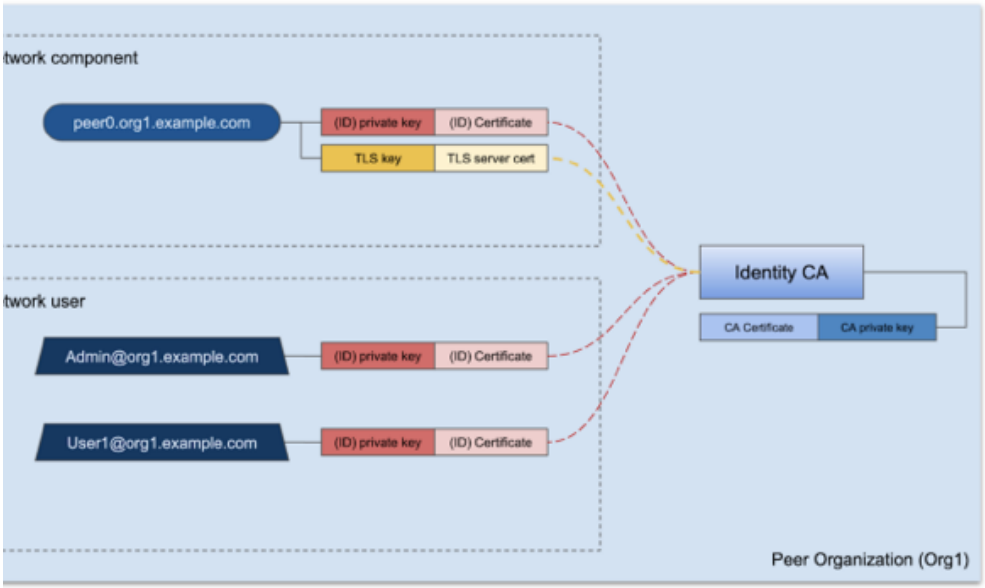
```
172-31-28-82:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.co
lsca/

 1 ubuntu ubuntu 806 May 11 03:22 tlsca.org1.example.com-cert.pem
172-31-28-82:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.co
 x509 -in tlsca/tlsca.org1.example.com-cert.pem -noout -subject -issuer
= US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
 US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
```

revious session about `cryptogen`, we see a separate TLS CA for all TLS l. When using a CA, the script of Test Network generates all crypto l for TLS with the same CA. There is no strict requirement whether ıe CA for each purpose or for both purposes.

ve see that only network component is equipped with TLS server ıtes, and network users (e.g. Admin, User1) is not. It works fine as tup only server authentication (one-way authentication) is required To know more about TLS settings in Hyperledger Fabric, refer to the ocumentation about this (link).

```
172-31-28-82:~/fabric-samples/test-network/organizations/peerOrganizations/org1.example.co
 x509 -in peers/peer0.org1.example.com/tls/server.crt -noout -subject -issuer
= US, ST = North Carolina, O = Hyperledger, OU = peer, CN = peer0
 US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
```

ıow it looks like in Test Network after launching it using CA Server.



Crypto material generated using `Fabric CA Server` in Test Network

CA Server

e seen the result. Now we can take a look on the CA Server side, and

ıe material inside CA Server.

ric-CA Server is being brought up as a docker container. The docker e file for it is **`./docker/docker-compose-ca.yaml`**. It brings up all the A servers. We take a look at the one for Org1.

```yaml
ca_org1:
  image: hyperledger/fabric-ca:$IMAGE_TAG
  environment:
    - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
    - FABRIC_CA_SERVER_CA_NAME=ca-org1
    - FABRIC_CA_SERVER_TLS_ENABLED=true
    - FABRIC_CA_SERVER_PORT=7054
  ports:
    - "7054:7054"
  command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
  volumes:
    - ../organizations/fabric-ca/org1:/etc/hyperledger/fabric-ca-server
  container_name: ca_org1
  networks:
    - test
```

The Org1 portion of docker-compose-ca.yaml configuration file

ɔservations in this configuration.

a predefined configuration file `fabric-ca-server-config.yaml` for ored in `organizations/fabric-ca/org1`. From the volumes (line 23–24) ɔped to the container. Therefore this file exists in the container when a-server is running. This is a practical way for managing ration, as we can edit in localhost while the edited configuration file ly in the container.

environment variables are set in the container. What is defined here rride the configuration file.

he command to bring up the Fabric CA Server is **`fabric-ca-server`** ith a bootstrap identity **`-b admin:adminpw`**. It will be used in the next en we enroll a CA Admin with Fabric-CA Client.

material is kept in **`/etc/hyperledger/fabric-ca-server`** when the
CA Server is running. We can examine the content in localhost,
ng to the mapping in the container (line 23–24).

```
172-31-28-82:~/fabric-samples/test-network$ ls -l organizations/fabric-ca/org1/

 1 root   root     843 May 11 03:22 IssuerPublicKey
 1 root   root     215 May 11 03:22 IssuerRevocationPublicKey
 1 root   root     806 May 11 03:22 ca-cert.pem
 1 ubuntu ubuntu 16044 Apr 11 03:52 fabric-ca-server-config.yaml
 1 root   root   61440 May 11 03:22 fabric-ca-server.db
 6 root   root    4096 May 11 03:22 msp
 1 root   root     932 May 11 03:22 tls-cert.pem
```

Material inside Fabric-CA Server, being mapped to localhost for inspection.

them we will see

: where the crypto material for CA of Org1 stored

`ert.pem` : the CA Certificate for Org1

`cert.pem` : the TLS server certificate for CA Server, used when
ssing the CA Server with TLS

`ic-ca-server.db` : the database of the Fabric CA Server, recording the
registration and certificate issuance

`erPublicKey` and `IssuerRevocationPublicKey` : the names are self-
anatory.

take a look at the CA Certificate **`ca-cert.pem`** for Org1.

```
172-31-28-82:~/fabric-samples/test-network$ openssl x509 -in organizations/fabric-ca/org1/
m -noout -subject -issuer
= US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
 US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
```

self-signed certificate. Again, this is just one option. With Fabric-CA
ve can incorporate an existing certificate (with its private key).

CA Server, as a server, supports TLS. Therefore a TLS server
ate is generated and kept as **`tls-cert.pem`**.

```
172-31-28-82:~/fabric-samples/test-network$ openssl x509 -in organizations/fabric-ca/org1/
em -noout -subject -issuer
= US, ST = North Carolina, L = Durham, O = org1.example.com, CN = 4bcb4020f578
 US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
```

s is just a TLS server certificate issued by CA of Org1.

responding private keys are kept in **`msp/keystore/`**.

```
72-31-28-82:~/fabric-samples/test-network$ ls -l organizations/fabric-ca/org1/msp/keystore/

1 root root 241 May 11 03:22 9f6b1edd099719b669c668b6f934ee543336bf2fc266c08305e6e635a425f00f_sk
1 root root 282 May 11 03:22 IssuerRevocationPrivateKey
1 root root  32 May 11 03:22 IssuerSecretKey
1 root root 241 May 11 03:22 ab4fedecfc905477f335b6d9113b1fdff3e300f26c766fc61d525f6601f91805_sk
```

ıot take too much effort to correlate the two secret keys matching the ificate for Org1 and TLS server certificate for the Fabric-CA Server.

cess in Detail

w the process executed in `network.sh` script. Here is the overall flow ating crypto material using CA Server. For an organization,

g up the Fabric-CA Server which is used as the CA of that nization.

Fabric-CA Client to enroll a CA Admin.

ı the CA Admin, use Fabric-CA Client to register and enroll every y (peer, orderer, user, etc) one by one to the Fabric-CA Server.

e the result material to the directory structure.

configuration is provided whenever needed. And repeat this process rganizations, as each of them has its own organization CA.

take Org1 as an example and take a look on the steps. And we will ne instruction in network.sh and perform this process once. This understand what happens behind.

clean up what has been deployed with,

```
st-network
work.sh down

/ou see permission error in removing some directories, perform
ollowing
m -r organizations/fabric-ca/org1/msp/
m -r organizations/fabric-ca/org2/msp/
m -r organizations/fabric-ca/ordererOrg/msp/
```

abric-CA Server

```
_TAG=latest docker-compose -f docker/docker-compose-ca.yaml up -
```

the CAs are up and running.

```
172-31-28-82:~/fabric-samples/test-network$ docker ps
ID        IMAGE                               COMMAND                  CREATED              STAT
      PORTS                                   NAMES
a3        hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   21 seconds ago       Up 1
      0.0.0.0:7054->7054/tcp                  ca_org1
a8        hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   21 seconds ago       Up 1
      7054/tcp, 0.0.0.0:9054->9054/tcp        ca_orderer
be        hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   21 seconds ago       Up 1
      7054/tcp, 0.0.0.0:8054->8054/tcp        ca_org2
```

nroll CA Admin

e using Fabric-CA Client to interact with the Fabric-CA Server, we d to specify the Fabric-CA Client home directory.

```
 -p organizations/peerOrganizations/org1.example.com/

t PATH=$PATH:/home/ubuntu/fabric-samples/bin
t
C_CA_CLIENT_HOME=${PWD}/organizations/peerOrganizations/org1.exa
com/
```

can enroll a CA Admin.

```
c-ca-client enroll -u https://admin:adminpw@localhost:7054 --
e ca-org1 --tls.certfiles ${PWD}/organizations/fabric-
g1/tls-cert.pem
```

nrollment,

strap identity `admin:adminpw` is specified

CA name is ca-org1

correct TLS server certificate for CA Server

```
172-31-28-82:~/fabric-samples/test-network$ fabric-ca-client enroll -u https://admin:admin
st:7054 --caname ca-org1 --tls.certfiles ${PWD}/organizations/fabric-ca/org1/tls-cert.pem
 08:18:38 [INFO] Created a default configuration file at /home/ubuntu/fabric-samples/test-
ganizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
 08:18:38 [INFO] TLS Enabled
 08:18:38 [INFO] generating key: &{A:ecdsa S:256}
 08:18:38 [INFO] encoded CSR
 08:18:38 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/test-network/org
/peerOrganizations/org1.example.com/msp/signcerts/cert.pem
 08:18:38 [INFO] Stored root CA certificate at /home/ubuntu/fabric-samples/test-network/or
s/peerOrganizations/org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
 08:18:38 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/test-network/orga
peerOrganizations/org1.example.com/msp/IssuerPublicKey
 08:18:38 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/test-n
anizations/peerOrganizations/org1.example.com/msp/IssuerRevocationPublicKey
```

r Fabric-CA Client has the credential of a CA Admin, and ready to
the enrollment for each entity. For information this credential is
he home directory (**`msp/keystore`** and **`msp/signcerts`**).

```
172-31-28-82:~/fabric-samples/test-network$ ls -l organizations/peerOrganizations/org1.exa
sp/keystore/

 1 ubuntu ubuntu 241 May 11 08:18 7f2635df932a1570d1fd19ed857893152e5e3536010ba6b8441ae1c1
k
172-31-28-82:~/fabric-samples/test-network$ ls -l organizations/peerOrganizations/org1.exa
sp/signcerts/

 1 ubuntu ubuntu 855 May 11 08:18 cert.pem
```

certificate is issued by the CA of Org1.

```
172-31-28-82:~/fabric-samples/test-network$ openssl x509 -in organizatio
ample.com/msp/signcerts/cert.pem -noout -subject -issuer
= US, ST = North Carolina, O = Hyperledger, OU = client, CN = admin
: US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
```

Register and Enroll Each Entity

1 entity (peer0, Admin and User), we will first register in the CA and then enroll the entity to obtain the credential.

l a **`config.yaml`** file to enable the OU identifiers, and keep the OU ers for each type of entity. They are **peer**, **orderer**, **client** and **admin**.

```
Js:
ole: true
entOUIdentifier:
ertificate: cacerts/localhost-7054-ca-org1.pem
rganizationalUnitIdentifier: client
rOUIdentifier:
ertificate: cacerts/localhost-7054-ca-org1.pem
rganizationalUnitIdentifier: peer
inOUIdentifier:
ertificate: cacerts/localhost-7054-ca-org1.pem
rganizationalUnitIdentifier: admin
ererOUIdentifier:
ertificate: cacerts/localhost-7054-ca-org1.pem
rganizationalUnitIdentifier: orderer
```

is file under client home directory:

ations/peerOrganizations/org1.example.com/msp/

just take **`peer0.org1.example.com`** as an example.

register this entity.

```
c-ca-client register --caname ca-org1 --id.name peer0 --
cret peer0pw --id.type peer --id.attrs
Registrar.Roles=peer"' --tls.certfiles
}/organizations/fabric-ca/org1/tls-cert.pem
```

egistration, besides the CA Name and TLS certificate of the CA
here are several items relevant to the identity of
rg1.example.com,

ame : peer0, this is the name of entity

ecret : peer0pw, this is the secret to be used when enrol this entity

ype : peer, here is where we specify the type of this entity. The correct
dentifier will be set according to the config.yaml above.

ttrs : “hf.Registrar.Roles=peer”

```
172-31-28-82:~/fabric-samples/test-network$ fabric-ca-client register --caname ca-org1 --i
r0 --id.secret peer0pw --id.type peer --id.attrs '"hf.Registrar.Roles=peer"' --tls.certfil
organizations/fabric-ca/org1/tls-cert.pem
 08:34:42 [INFO] Configuration file location: /home/ubuntu/fabric-samples/test-network/org
/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
 08:34:42 [INFO] TLS Enabled
 08:34:42 [INFO] TLS Enabled
peer0pw
```

ectively puts the information into the Fabric-CA Server database for
rollment. Nothing is received after registration.

e can use the name (`id.name`) and secret (`id.secret`) to enrol the
rg1.example.com, with this command.

```
c-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --
e ca-org1 -M
}/organizations/peerOrganizations/org1.example.com/peers/peer0.o
xample.com/msp --csr.hosts peer0.org1.example.com --
ertfiles ${PWD}/organizations/fabric-ca/org1/tls-cert.pem
```

the CA Name and TLS certificate of the CA Server, we need to specify

URL with entity’s name and secret, in the format as `https://<name>:
et>@<CA’s URL>`

MSP location (local directory) to store the crypto material received

ıost in the default CSR (in `fabric-ca-client.yaml`) is changed to **`0.org1.example.com`**. This is to override the default setting.

```
172-31-28-82:~/fabric-samples/test-network$ fabric-ca-client enroll -u https://peer0:peer0
st:7054 --caname ca-org1 -M ${PWD}/organizations/peerOrganizations/org1.example.com/peers/
.example.com/msp --csr.hosts peer0.org1.example.com --tls.certfiles ${PWD}/organizations/f
rg1/tls-cert.pem
 08:56:20 [INFO] TLS Enabled
 08:56:20 [INFO] generating key: &{A:ecdsa S:256}
 08:56:20 [INFO] encoded CSR
 08:56:20 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/test-network/org
/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/signcerts/cert.pem
 08:56:20 [INFO] Stored root CA certificate at /home/ubuntu/fabric-samples/test-network/or
s/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/cacerts/localhost-70
.pem
 08:56:20 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/test-network/orga
peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/IssuerPublicKey
 08:56:20 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/test-n
anizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/IssuerRevoc
cKey
```

e crypto material for peer0.org1.example.com is stored in **ations/peerOrganizations/org1.example.com/peers/peer0.org1.example.c** , as specified in the enroll command.

```
172-31-28-82:~/fabric-samples/test-network$ ls -l organizations/peerOrganizations/org1.exa
eers/peer0.org1.example.com/msp/

 1 ubuntu ubuntu  843 May 11 08:56 IssuerPublicKey
 1 ubuntu ubuntu  215 May 11 08:56 IssuerRevocationPublicKey
 2 ubuntu ubuntu 4096 May 11 08:56 cacerts
 2 ubuntu ubuntu 4096 May 11 08:56 keystore
 2 ubuntu ubuntu 4096 May 11 08:56 signcerts
 2 ubuntu ubuntu 4096 May 11 08:56 user
```

ectory is familiar, as it is the same directory structure we observe Here we just show the certificate (inside **`signcerts/`**).

```
172-31-28-82:~/fabric-samples/test-network$ openssl x509 -in organizations/peerOrganizations/o
e.com/peers/peer0.org1.example.com/msp/signcerts/cert.pem -noout -text
e:

rsion: 3 (0x2)
rial Number:
  6c:d5:8c:b7:2e:2c:ac:dd:48:aa:9c:a6:26:a1:fd:ee:48:a5:4c:50
gnature Algorithm: ecdsa-with-SHA256
suer: C = US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
lidity
  Not Before: May 11 08:51:00 2020 GMT
  Not After : May 11 08:56:00 2021 GMT
bject: C = US, ST = North Carolina, O = Hyperledger, OU = peer, CN = peer0
bject Public Key Info:
  Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
          04:e3:e1:2d:81:5e:d8:db:7e:22:14:a7:0f:87:dc:
          3e:32:31:c2:12:ac:e3:75:e3:dd:97:b4:af:a8:b6:
          64:a1:0c:c1:5b:d3:bc:0f:f5:97:7c:39:dc:00:99:
          8d:d1:c7:6d:4d:43:e5:76:4a:1c:46:33:d3:e6:0d:
          5e:8e:fe:be:a1
      ASN1 OID: prime256v1
      NIST CURVE: P-256
09v3 extensions:
  X509v3 Key Usage: critical
      Digital Signature
  X509v3 Basic Constraints: critical
      CA:FALSE
  X509v3 Subject Key Identifier:
      C6:6A:D9:0A:C1:C9:83:DB:79:97:5A:7D:A6:22:67:C7:0B:0A:D0:69
  X509v3 Authority Key Identifier:
      keyid:6D:5D:D3:04:0B:F4:6E:18:D9:6C:D9:D1:A1:15:E6:75:5E:BC:26:AD

  X509v3 Subject Alternative Name:
      DNS:peer0.org1.example.com
  1.2.3.4.5.6.7.8.1:
      {"attrs":{"hf.Affiliation":"","hf.EnrollmentID":"peer0","hf.Type":"peer"}}
```

the subject and issuer, we also see some more information about the ıte of **`peer0.org1.example.com`**. Take a note on the **X509v3 Subject tive Name** with **`peer0.org1.example.com`**, and also the attributes d in registration and enrollment.

need to enrol a TLS server cert.

```
c-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --
e ca-org1 -M
}/organizations/peerOrganizations/org1.example.com/peers/peer0.o
xample.com/tls --enrollment.profile tls --csr.hosts
.org1.example.com --csr.hosts localhost --tls.certfiles
}/organizations/fabric-ca/org1/tls-cert.pem
```

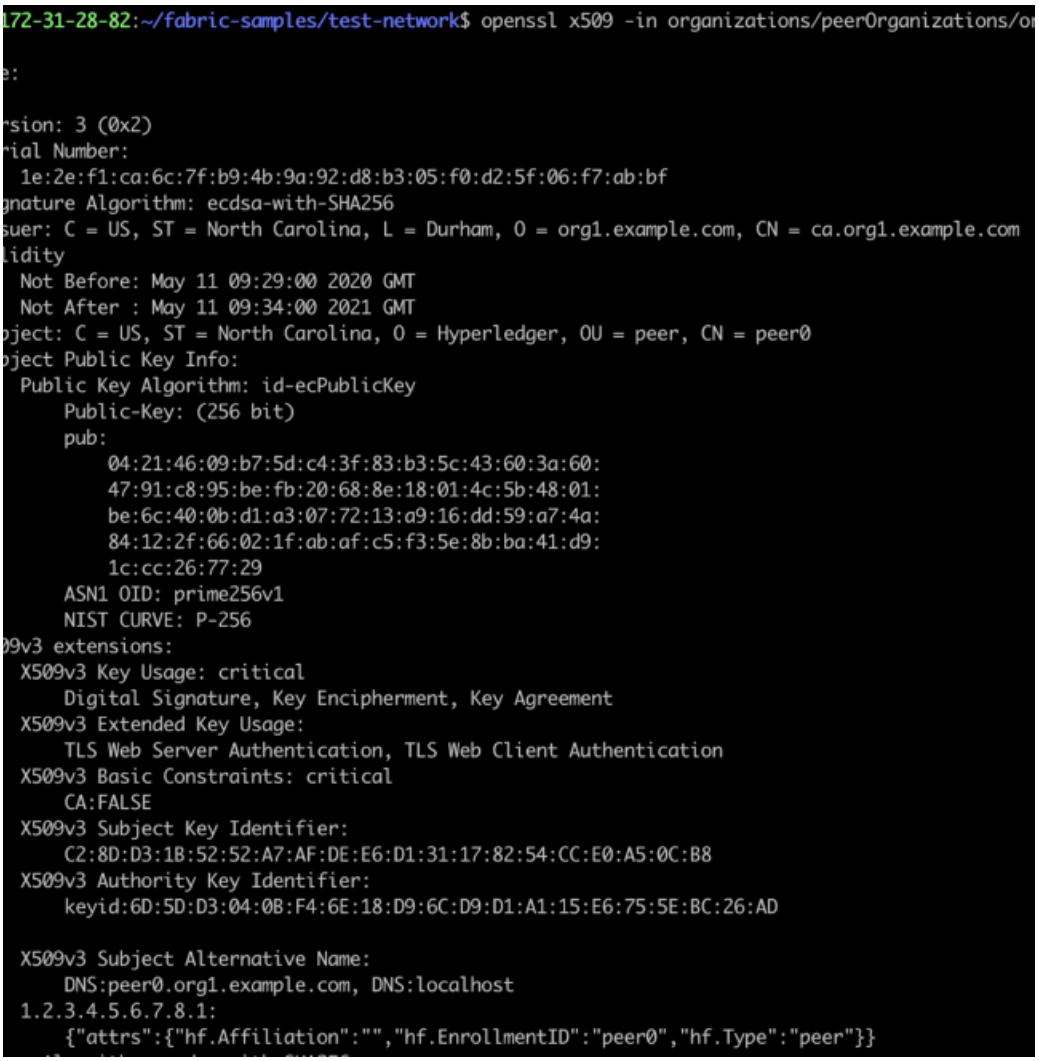
ɪmand is similar to enrol the credential for identity. The only
.ce is that the `enrollment.profile` is `tls`, and CSR hosts.

```
172-31-28-82:~/fabric-samples/test-network$ fabric-ca-client enroll -u https://peer0:peer0
st:7054 --caname ca-org1 -M ${PWD}/organizations/peerOrganizations/org1.example.com/peers/
.example.com/tls --enrollment.profile tls --csr.hosts peer0.org1.example.com --csr.hosts l
-tls.certfiles ${PWD}/organizations/fabric-ca/org1/tls-cert.pem
 09:33:31 [INFO] TLS Enabled
 09:33:31 [INFO] generating key: &{A:ecdsa S:256}
 09:33:31 [INFO] encoded CSR
 09:33:31 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/test-network/org
/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/signcerts/cert.pem
 09:33:31 [INFO] Stored TLS root CA certificate at /home/ubuntu/fabric-samples/test-networ
tions/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/tlscacerts/tls-l
054-ca-org1.pem
 09:33:31 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/test-network/orga
peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/IssuerPublicKey
 09:33:31 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/test-n
anizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/IssuerRevoc
cKey
```

ılt is stored in
ɜtions/peerOrganizations/org1.example.com/peers/peer0.org1.example.c
, as specified in the enroll command.

```
172-31-28-82:~/fabric-samples/test-network$ ls -l organizations/peerOrganizations/org1.exa
eers/peer0.org1.example.com/tls/

 1 ubuntu ubuntu  843 May 11 09:33 IssuerPublicKey
 1 ubuntu ubuntu  215 May 11 09:33 IssuerRevocationPublicKey
 2 ubuntu ubuntu 4096 May 11 09:33 cacerts
 2 ubuntu ubuntu 4096 May 11 09:33 keystore
 2 ubuntu ubuntu 4096 May 11 09:33 signcerts
 2 ubuntu ubuntu 4096 May 11 09:33 tlscacerts
 2 ubuntu ubuntu 4096 May 11 09:33 user
```

e just see the TLS server certificate (in **`signcerts/`**).



e X509v3 Extended Key Usage. As we specify enrollment profile as
see **TLS Web Server Authentication** and **TLS Web Client**
ication here.

or **`peer0.org1.example.com`**. The process is repeated for **Admin** and
Note that the `network.sh` script does not enrol the TLS client
ate for them.

Move material to the right directory

ectory of identity is kept inside `<entity>/msp/`, and the structure is
use in docker-compose files. However the directory of
`>/tls/` is not the format. Therefore we need to move TLS CA (inside

`rts/`), server cert (inside `signcerts/`) and server key (inside `e/`) out and fit them into the `<entity>/tls/` with proper names.

by the CA certificates, we can reconstruct the desired directory e. This is good enough for bringing up containers with docker-e.

read the remaining steps in **`network.sh`** and **`registerEnroll.sh`** script lete the whole process.

ary

rticle, we use Test Network to show the two different ways of ion crypto material for a consortium network. While `cryptogen` is a easy way with minimum configuration to bring up a workable t, it has certain limitations and therefore is always considered for purposes. A more practical way for production is to bring up a CA nd generate the material according to one's need in a consortium t. In the Test Network, we see how Fabric-CA Server is used as the CA and how to use Fabric-CA Client to perform the whole process.

dger Fabric Certificate Authority Permissioned Blockchains Identity