Project Report

Defence Research and Development Organisation
IGDTUW

Name:	Sugandh,Nitisha Sharma,Priya Agrasen
Student Number:	SAG GROUP 1
Research Centre:	DRDO
Research Project Title:	Hyperledger Fabric Project
Primary Supervisor:	Mr. Girish Mishra

---

1 Initial Summary

The development environment was configured using Windows Subsystem for Linux (WSL) to establish a Hyperledger Fabric network. The official GitHub repository of Hyperledger Fabric was accessed, and the required scripts were downloaded using:

```
wget https://raw.githubusercontent.com/hyperledger/fabric/v2.5.0/scripts/bootstrap.sh
chmod +x bootstrap.sh
./bootstrap.sh
```

The path to the Fabric binaries was appended to the shell configuration file using the following command in the terminal:

```
nano ~/.bashrc
# Add the following line at the end of the file:
export PATH=$PATH:/path/to/fabric-samples/bin
```

The network lifecycle was managed through the `network.sh` script, which is part of the downloaded Fabric samples.

The network was brought up and a channel was created using the following command:

```
./network.sh up createChannel -ca
```

To terminate the network and remove associated containers and configurations, the following command was used:

```
./network.sh down
```

Docker and Docker Compose were installed and configured to support the containerized infrastructure required by the Fabric framework.

Chaincode deployment was executed using the command below, where parameters were customized as per the project requirements. This listing environment ensures long lines wrap correctly within the document margins:

```
./network.sh deployCC -ccn <chaincode_name> -ccp <path_to_chaincode> -ccl <language> -ccv 1 -ccep <
    endorsement_policy>
```

This ensured that the chaincode was packaged, installed, approved, and committed to the specified channel, enabling ledger operations and smart contract interactions.

This section should be approximately 0.5 – 1.0 pages long, and can include references if necessary. You can add both textual and parenthetical references as appropriate — e.g., **?** published an article with very rapid citation rates, but is gradually becoming eclipsed from competition in recent years (e.g., **???**, to name but a few).

Goal 1: Identity Enrollment and Chaincode Access

The objective is to enroll the administrator and application user identities on the Hyperledger Fabric test network to enable secure interaction with the deployed chaincode through a user interface (UI) layer. This enrollment process ensures proper identity management and access control, facilitating authenticated operations from the client-side application.

2 Identity Management and Chaincode Integration

To enable interaction between the UI and the deployed chaincode on the Hyperledger Fabric test network, identity enrollment scripts and a chaincode access interface were developed.

2.1 Scripts and Their Functionality

The system architecture includes three key Node.js scripts:

- **enrollAdmin.js** – Registers and enrolls the network administrator using the Fabric Certificate Authority (CA). The admin identity is stored in a local wallet to manage network-level privileges.
- **enrollUser.js** – Registers and enrolls a standard application user identity under the already enrolled administrator. The user identity is stored in the wallet with limited permissions suitable for invoking and querying chaincode functions.
- **menu.js** – Serves as the terminal-based UI to interact with the deployed chaincode. It connects to the blockchain network using the identity stored in the wallet and allows users to execute defined smart contract functions.

This setup ensures that user credentials are securely stored and accessible for invoking smart contract logic through the interface.

2.2 Progress and Functional Outcomes

Successful execution of query commands via 'menu.js' confirmed proper identity-based network access and successful chaincode connection. The scripts were able to perform read-only operations on the ledger without error.

2.3 Invoke Errors and Encountered Limitations

While query operations were successful, invoke operations requiring endorsements and ledger updates failed. The following error was observed during such transactions:

```
DiscoveryService: certificate error: failed constructing descriptor for chaincodes:<name:"certificate
    ">
```

This error indicates a potential issue in the discovery mechanism of the Fabric network, likely tied to TLS misconfiguration, peer misidentification, or incomplete chaincode metadata propagation.

2.4 Troubleshooting Efforts

As part of the debugging process, the environment variables for both organizations was rechecked and sourced .Path of certificate contract in enrollAdmin.js,enrollUser.js and menu.js was checked .

2.5 Resolution and Goal Achievement

Upon investigation, the root cause of the chaincode discovery error was traced to a missing export statement in the main chaincode entry file `index.js`. Without this export, the network was unable to correctly identify and reference the smart contract class during discovery and invocation procedures.

The following line was added at the end of `index.js` to properly expose the chaincode class to the Fabric network:

```
module.exports.CertificateContract = CertificateContract;
```

This change ensured that the chaincode definition was fully accessible during service discovery, resolving the previously encountered error:

```
DiscoveryService: certificate error: failed constructing descriptor for chaincodes:<name:"certificate
    ">
```

After implementing this fix, the terminal interface `menu.js` was successfully able to perform invoke operations, including writing to the ledger, in addition to previously functional query operations.

Conclusion: With this correction, authenticated interaction between the user interface and the chaincode was fully enabled. Thus, the objective defined under **Goal 1: Identity Enrollment and Chaincode Access** has been successfully achieved.