Setup Instructions

Requirements: 2 Machines running CentOS 7 (can be virtual machines)

The machine running MultiChain and acting as the VPN server is going to be a Main Station machine. The second machine acting as a VPN client will be a Voting Machine.

Install MultiChain

On the first system:

1. Download and install Multichain to allow command line functionality

```
su
cd /tmp
wget https://www.multichain.com/download/multichain-latest.tar.gz
tar -xvzf multichain-latest.tar.gz
cd multichain-1.0.1
mv multichaind multichain-cli multichain-util /usr/local/bin
exit
```

2. Create a new blockchain through command line

```
multichain-util create "blockChainName"
```

3. Change default parameters of blockchain parameter file

```
su
vim ~/.multichain/"blockChainName"/params.dat
```

   Change the following:

```
anyone-can-connect=true
 anyone-can-send=true
```

4. Change default parameters of blockchain config file

```
su
vim ~/.multichain/"blockChainName"/multichain.conf
```

   Write down rpcuser/rpcpassword or change them to something easy to remember add this line to config file, which allows any ip to connect to chain (only for development, should be edited before deployment to block unauthorised connections):

```
rpcallowip=0.0.0.0/0
```

   Your conf file should have three lines, rpcuser, rpcpassword and rpcallowip.

5. Initialise the blockchain

```
multichaind "blockChainName" -daemon
```

Now that your blockchain is up you can leave this system idle.

Install Mono and Monodevelop IDE

This needs to be done on both machines

1. Add the Mono repository to your system

```
yum install yum-utils
rpm --import "http://keyserver.ubuntu.com/pks/lookup?op=get&search=
      0x3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF"
yum-config-manager --add-repo
      http://download.mono-project.com/repo/centos7/
```

2. Install Mono

```
su
yum install mono-devel mono-complete
exit
```

3. Verify Mono installation

   After the installation has completed successfully, it is a good idea to run through the basic hello world programs outlined here:

   http://www.mono-project.com/docs/getting-started/mono-basics/

   For the context of this program, the Gtk# test is especially important

4. Install Monodevelop IDE

```
su
yum install monodevelop monodevelop-database monodevelop-nunit
exit
```

Install OpenVPN

(Source: https://www.howtoforge.com/tutorial/how-to-install-openvpn-on-centos-7/)

On the first system (this will be our VPN server):

1. Enable the EPEL repository

```
su
yum -y install epel-release
```

2. Install OpenVPN and Easy RSA

```
yum -y install easy-rsa openvpn
```

3. Configure Easy RSA

   We will be generating some keys and certificates here.

   a. Copy easy-rsa script generation to “/etc/openvpn/”.

```
cp -r /usr/share/easy-rsa/ /etc/openvpn/
```

b. Go to easy-rsa directory and edit the default values in the vars file

```
cd /etc/openvpn/easy-rsa/2.0/
vim vars
```

We recommend a key size of 2048 to be used for this VPN.

c. Use vars file to generate new keys for our installation and clean existing keys

```
source ./vars
./clean-all
```

d. Build a new certificate authority (ca).

```
./build-ca
```

You will be asked to enter details (Country Name, etc.) before new ca is generated, enter details applicable to you. Leave the password and optional company name blank. Once finished, this will create two files (ca.crt and ca.key) in /etc/openvpn/easy-rsa/2.0/keys/

e. Generate server key and certificate

```
./build-key-server server
```

Again, you will be prompted for details. These should be similar to the details you entered in the last step.

f. Build a Diffie-Hellman key exchange

```
./build-dh
```

This will take some time to generate especially on larger key sizes

g. Generate client key and certificate

```
./build-key client
```

Enter your details again bearing in mind that these are client keys, not server keys. It is recommended to change the common name to reflect this difference

h. Copy the “keys/” directory to “/etc/openvpn/”

```
cd /etc/openvpn/easy-rsa/2.0/
cp -r keys/ /etc/openvpn/
```

4. Configure OpenVPN
   a. Create configuration file

```
cd /etc/openvpn/
vim server.conf
```

Copy configuration below:

```
#OpenVPN port number
port 1194

#You can use udp or tcp
proto udp

# "dev tun" will create a routed IP tunnel.
dev tun

#Certificate Configuration

#ca certificate
ca /etc/openvpn/keys/ca.crt

#Server Certificate
cert /etc/openvpn/keys/server.crt

#Server Key and keep this is secret
key /etc/openvpn/keys/server.key

#See the size a dh key in /etc/openvpn/keys/
dh /etc/openvpn/keys/dh1024.pem

#Internal IP will get when already connect
server 192.168.200.0 255.255.255.0

#this line will redirect all traffic through our OpenVPN
push "redirect-gateway def1"

#Provide DNS servers to the client, you can use Google DNS
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"

#Enable multiple client to connect with same key
duplicate-cn

keepalive 20 60
comp-lzo
persist-key
persist-tun
daemon

#enable log
log-append /var/log/openvpn-status.log

#Log Level
verb 3
```

b. Create a log file for the VPN, if you've specified a different file for logging use that path instead.

```
touch /var/log/openvpn-status.log
```

5. Configure firewalld to allow OpenVPN service
    a. List current services
    ```
    firewall-cmd --list-services
    ```

    b. Add OpenVPN to services list
    ```
    firewall-cmd --add-service openvpn
    ```

    Run the list-services command again to verify OpenVPN has been added

    c. This change won't be permanent between restarts, to fix this run
    ```
    firewall-cmd --permanent --add-service openvpn
    ```

    d. Add a masquerade
    ```
    firewall-cmd --add-masquerade
    ```

    e. Again, this change will not be permanent until we run
    ```
    firewall-cmd --permanent --add-masquerade
    ```

    To verify the masquerade has been added successfully run
    ```
    firewall-cmd --query-masquerade
    ```

    This should return a "yes" if the masquerade exists
6. Start the server by running
```
systemctl enable openvpn@service.service
systemctl start openvpn@server.service
systemctl status -l openvpn@server.service
```

On second machine (this will be our VPN client machine)

1. Connect to the server we just created and download these three files we created earlier using scp:
    - Ca.crt
    - Client.crt
    - client.key
2. Create a new file called "client.ovpn" and add the following:
```
client
dev tun
proto udp

#Server IP and Port
remote 192.168.1.104 1337

resolv-retry infinite
nobind
```

```
persist-key
persist-tun
mute-replay-warnings
ca ca.crt
cert client.crt
key client.key
ns-cert-type server
comp-lzo
```

3. Install openvpn in the same way as in step 1 and 2 in the previous section (no need to install easy-rsa)
4. Run OpenVPN as follows

```
sudo openvpn --config client.ovpn
```

Install VotingMultiChain

1. Start MultiChain on Main Station by running

```
multichaind “blockChainName” -daemon
```

2. Clone this repository using git
3. Open cloned repository in Monodevelop in order to compile and run the program
4. Enter blockchain details into the "Connect to blockchain window".

   These details can be seen on your other systems param and conf file,

   and the output when you initially connected to the chain.

   For example given this

   multichaind blockChainName@192.168.218.131:4391

   IP: 192.168.218.131

   Port: 4390 (RPCPort is always -1 than generic port)

   RPCUsername/RPCPassword (What was in your .conf file)

   ChainName: blockChainName
5. If the details entered are correct, the program will connect successfully and show a login window
6. Voting Stations just need to compile and run the software with the correct RPC information.

NOTE: Connection information required in step 4 is currently hard-coded in MainWindow.cs