Appendix G

PostgreSQL SSL Configuration

These instructions will guide you through the process of configuring PostgreSQL to use SSL for secure connections. An intermediate CA will be placed in the chain of trust. While this is not strictly necessary, it is a good idea as it keeps the root CA safe (i.e., offline) once the intermediate certificates have been created. In the case of a security breach, only the intermediate certificate needs to be revoked.

This will show how to create a self-signed root CA for the purposes of demonstration, but feel free to substitute your own root CA:
Creating certificates
Configuring openssl.cnf

These instructions expect that your openssl configuration file is located at /etc/ssl/openssl.cnf.
From a default configuration, ensure that the following line in the [v3_ca] section has been uncommented:

keyUsage = cRLSign, keyCertSign

Create a Self-Signed CA (Optional)

You will likely have a certificate signed by a trusted CA, but for some installations, or to just try out these instructions, you may want to create a self-signed certificate.

```
Create a private key:
openssl genrsa -aes256 -out ca.key 4096
```

You will be required to enter a passphrase. This should be long and guarded well.

```
Create a self-signed certificate:
openssl req -new -x509 -sha256 -days 1825 -key ca.key -out ca.crt \
-subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data Solutions/CN=root-ca"
```

Create the Intermediate CAs

Now that the root CA is worked out, create the intermediate CAs that will be used to sign server and client certificates.

```
Create the server intermediate private key;
openssl genrsa -aes256 -out server-intermediate.key 4096
```

You will be required to enter a passphrase. It is best not to reuse the passphrase from your root key.

```
Create the server intermediate certificate signing request (CSR):
openssl req -new -sha256 -days 1825 -key server-intermediate.key -out server-intermediate.csr \
-subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data Solutions/CN=server-im-ca"
```

```
Create the server intermediate certificate by signing with the CA certificate:
openssl x509 -extfile /etc/ssl/openssl.cnf -extensions v3_ca -req -days 1825 \
-CA ca.crt -CAkey ca.key -CAcreateserial \
-in server-intermediate.csr -out server-intermediate.crt
```

Repeat the process to create the client intermediate CA:

```
openssl genrsa -aes256 -out client-intermediate.key 4096
openssl req -new -sha256 -days 1825 -key client-intermediate.key -out client-intermediate.csr \
-subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data Solutions/CN=client-im-ca"
openssl x509 -extfile /etc/ssl/openssl.cnf -extensions v3_ca -req -days 1825 \
-CA ca.crt -CAkey ca.key -CAcreateserial \
-in client-intermediate.csr -out client-intermediate.crt
```

Create Server/Client Certificate

Server and client certificates are signed by their respective intermediate CAs rather than the root CA. Additionally, the common name on server certificates must match the hostname of the server, and the common name of the client certificates must match the client's PostgreSQL user logon (or be mapped in pg_ident.conf). The private keys will be created without passphrases to allow automatic startup of the PostgreSQL server and client.

```
Create a server certificate:
openssl req -nodes -new -newkey rsa:4096 -sha256 -keyout server.key -out server.csr \
-subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data Solutions/CN=server.crunchydata.com"
openssl x509 -extfile /etc/ssl/openssl.cnf -extensions usr_cert -req -days 1825 \
-CA server-intermediate.crt -CAkey server-intermediate.key \
-CAcreateserial -in server.csr -out server.crt
```

```
Create a client certificate:
openssl req -nodes -new -newkey rsa:4096 -sha256 -keyout client.key -out client.csr \
-subj "/C=US/ST=VA/L=Arlington/O=Crunchy Data Solutions/CN=pgusername"
openssl x509 -extfile /etc/ssl/openssl.cnf -extensions usr_cert -req -days 1825 \
-CA client-intermediate.crt -CAkey client-intermediate.key \
-CAcreateserial -in client.csr -out client.crt
Configuring PostgreSQL
Server Configuration
```

The examples below will use /var/lib/pgsql/12/data as the data_directory setting in postgresql.conf.

```
Copy the root CA
cp ca.crt /var/lib/pgsql/12/data/ca.crt
```

```
Copy the server key
cp server.key /var/lib/pgsql/12/data/server.key
```

The server, server-intermediate, and root ca certificates need to be copied to PostgreSQL's server.crt.

```
The exact order specified here is required:
cat server.crt server-intermediate.crt ca.crt > /var/lib/pgsql/12/data/server.crt
```

Set permissions (this is required for server start).

```
chown postgres:postgres /var/lib/pgsql/12/data/ca.crt \
/var/lib/pgsql/12/data/server.crt /var/lib/pgsql/12/data/server.key
chmod 600 /var/lib/pgsql/12/data/ca.crt /var/lib/pgsql/12/data/server.crt \
/var/lib/pgsql/12/data/server.key
```

Configure /var/lib/pgsql/12/data/postgresql.conf with the SSL settings:

```
ssl = on
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
```

```
ssl_ca_file = 'ca.crt'
```

Ensure that pg_hba.conf requires certs for the clients if you do not want them to be optional:

```
hostssl all all network/mask cert
```

Restart the server for settings to take effect.

Client Configuration

The examples below assume you are logged on to the OS as the user you want to configure.

Copy the root CA.

```
cp ca.crt ~/.postgresql/root.crt
```

Copy the client key.

```
cp client.key ~/.postgresql/postgresql.key
```

The client, client-intermediate, and root ca certificates need to be copied to the client's postgresql.crt. The exact order specified here is required:

```
cat client.crt client-intermediate.crt ca.crt > ~/.postgresql/postgresql.crt
```

Set permissions (this is required for client operation).

```
chmod 600 \
~/.postgresql/root.crt \
~/.postgresql/postgresql.key \
~/.postgresql/postgresql.crt
```

Note that these files can also be configured with environment variables. See http://www.postgresql.org/docs/current/static/libpq-ssl.html for more information.

Running the Client

When running client software, it is best to use the verify-full ssl mode. See the link in Client Configuration for a description of what the ssl modes mean and what level of protection they provide.

An example using psql:

```
psql "postgresql://server.crunchydata.com/postgres?sslmode=verify-full"
```