# Simplify SQL*Net encryption

**Thomas Lehmann**
**Robotron Datenbank-Software GmbH**
**01189 Dresden**

**tags**
oracle, sql*net, encryption, public key, private key, ssl, tsl, cipher suite, AES256, data integrity

**introduction**
Oracle Databases beginning to move from on premise to the cloud. From that point of view network encryption is one of the most important thing. When you look into the Oracle Database Licensing Information you'll see that Native Network Encryption and TCP/IP with SSL/TLS are no longer part of the Advanced Security Option. That means you can use that feature for free in all database editions.

**Native Network Encryption**
The algorithm behind is the so called Diffie-Hellman key negotiation algorithm. Both sessions share non secret information and calculate a secret based on them that nobody other knows.
The Native Network Encryption is very easy to implement. You need to configure it in sqlnet.ora on client and server side.

Possible values to enable encryption and check summing are rejected, accepted, requested and required. The settings can be done on client and server side.

| | | Server Setting | | | |
|---|---|---|---|---|---|
| | | **REJECTED** | **ACCEPTED** | **REQUESTED** | **REQUIRED** |
| Client Setting | **REJECTED** | OFF | OFF | OFF | FAIL |
| | **ACCEPTED** | OFF | OFF (default) | ON | ON |
| | **REQUESTED** | OFF | ON | ON | ON |
| | **REQUIRED** | FAIL | ON | ON | ON |

In a second parameter you have to specify the encryption algorithm. In Oracle 12c that can be AES128, AES192 or AES256.

A valid setup for encryption could be the following:

```
sqlnet.encryption_server=required
sqlnet.encryption_types_server=(AES256)

sqlnet.encryption_client=requested
sqlnet.encryption_types_client=(AES256)
```

In addition to the encryption of network traffic it's possible to check the data integrity with a check summing mechanism. That could be setup in sqlnet.ora as well.

```
sqlnet.crypto_checksum_server=requested
sqlnet.crypto_checksum_types_server = (SHA256)

sqlnet.crypto_checksum_client=requested
sqlnet.crypto_checksum_types_client = (SHA256)
```

**SSL/TLS Encryption**
The SSL / TLS encryption is the industrial standard for data encryption. It's based on a public/private key infrastructure. To implement that kind of data encryption you need to setup an oracle wallet and exchange the keys between the database server and the clients. You also need to configure the supported cipher suites on both sides.

Create Wallet and self-signed certificates on server side:

```
$ orapki wallet create -wallet ./server_wallet -auto_login -pwd server01

$ orapki wallet add -wallet ./server_wallet -dn "CN=server" -keysize 1024
-self_signed -validity 365 -pwd server01

$ orapki wallet display -wallet ./server_wallet
Requested Certificates:
User Certificates:
Subject:        CN=server
Trusted Certificates:
Subject:        CN=server

$ orapki wallet export -wallet ./server_wallet -dn "CN=server" -cert
./server_wallet/cert.txt
```

Do the same on client side:

```
orapki wallet create -wallet ./client_wallet -auto_login -pwd client01

orapki wallet add -wallet ./client_wallet -dn "CN=client" -keysize 1024 -
self_signed -validity 365 -pwd client01

orapki wallet export -wallet ./client_wallet -dn "CN=client" -cert
./client_wallet/cert.txt

Exchange certificates and import to wallet:
$ orapki wallet add -wallet ./server_wallet -trusted_cert -cert cert.txt
-pwd server01

orapki wallet add -wallet ./client_wallet -trusted_cert -cert cert.txt -
pwd client01
```

After that we need to configure the database listener to support SSL connections.
On server-side modify listener.ora and sqlnet.ora.

Enable TCPS protocol in listener.ora, define cipher suites and wallet location

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC2)))
    (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST  =
server01.robotron.de)(PORT = 1521)))
    (DESCRIPTION = (ADDRESS = (PROTOCOL = TCPS)(HOST =
server01.robotron.de)(PORT = 2484)))
  )

SSL_CLIENT_AUTHENTICATION = FALSE
SSL_CIPHER_SUITES= (SSL_RSA_WITH_AES_128_CBC_SHA,
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA)
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE)
     (METHOD_DATA = (DIRECTORY = /home/oracle/Wallets/server_wallet))
   )
```

Modify sqlnet.ora and include encryption information

```
SQLNET.AUTHENTICATION_SERVICES=(TCPS,NTS)
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_VERSION = 1.2
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE)
     (METHOD_DATA = (DIRECTORY = /home/oracle/Wallets/server_wallet))
   )
SSL_CIPHER_SUITES= (SSL_RSA_WITH_AES_128_CBC_SHA,
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA)
```

Restart Listener and check status

```
lsrnctl stop
lsrnctl start
lsrnctl status
```

That’s all for the server side configuration.
Now we need to edit sqlnet.ora on client side to enable ssl:

```
SQLNET.AUTHENTICATION_SERVICES=(TCPS,NTS)
SSL_CLIENT_AUTHENTICATION = FALSE
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE)
     (METHOD_DATA = (DIRECTORY = C:\Oracle\Wallets\client_wallet))
   )

SSL_CIPHER_SUITES= (SSL_RSA_WITH_AES_128_CBC_SHA,
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA)
```

With that setup you should be able to open a secure sqlnet connection. It's worth to check with a network tool that the configuration works as expected.

**contact:**
Thomas Lehmann
Robotron Datenbank-Software GmbH
Stuttgarter Straße 29
D-01189 Dresden

| | |
|---|---|
| Telefon: | +49 (0) 351-25859 2782 |
| Fax: | +49 (0) 351-25859 3696 |
| E-Mail | thomas.lehmann@robotron.de |
| Internet: | www.robotron.de |