Configuring Secure Communication

Prime Cable Provisioning secures all TCP based interactions through the use of Secure Socket Layer (SSL) protocol. SSL is a standard based protocol that enables secure communication. Prime Cable Provisioning supports TLS 1.2 by default.

SSL protects the following interactions:

- Clients using the Prime Cable Provisioning API to interact with the RDU.
- DPE and RDU interactions.
- CPNR-EP and RDU interactions.
- Client interaction and RDU interaction with the new web services interface.

Certain SSL property configuration can be done using the changeSSLProperties.sh tool. For details about the tool, see Using changeSSLProperties.sh.

- Key and Certificate Management, on page 1

Key and Certificate Management

The RDU stores the certificates that the SSL protocol requires for authentication in a keystore. This keystore is a file that stores cryptographic keys and certificates. The keystore is generated with the help of a tool available with the JRE called the keytool. The generated certificates are validated for SSL communication before establishing the SSL socket. The keystore files are stored under BPR_HOME/lib/security folder by default. The keystore location is configurable in Prime Cable Provisioning. The bprAgent must be restarted after changing the default keystore location.

You can use Prime Cable Provisioning to configure the server certificate keystore and the cacerts keystore by using the keytool utility. The keytool is a key and certificate-management utility, which you use to administer the certificates on the RDU server and the clients. The keytool utility resides in the Prime Cable Provisioning default installation directory, at BPR_HOME/jre/bin/keytool.

Note You must execute the keytool utility bundled with this Prime Cable Provisioning version, because the keystore file format varies between keytool releases.

There are two keystores on the RDU server. The keystores are the cacerts keystore and the server certificates keystore.

- The cacerts keystore contains public key certificates that the components trust for authenticating the server certificates.
- The server certificates keystore contains the private key and the associated certificate chain for the server-side certificate, which is used to authenticate the clients.

All component SSL services share a single cacerts keystore. This keystore can contain any number of signing authority certificates. The name of the cacerts keystore is fixed, and it must always reside in *BPR_HOME/jre/lib/security* directory. Prime Cable Provisioning ships with a default cacerts keystore, which can be manipulated by adding and removing signing authority certificates.

Configuring SSL Post Installation

While installing Prime Cable Provisioning, you are asked if you want to configure SSL (secure-mode communication). If you choose yes, the SSL mode is enabled and you are prompted to enter the RDU certificate details. In case you choose no or for some reason, SSL does not get enabled, no RDU certificate is created.

This section explains how to configure SSL in case you have not done it during the installation.

Configuring SSL on RDU

You can use the -ssl option to enable or disable SSL on RDU.

To enable SSL on the RDU server:

Step 1 Using the keytool utility changeSSLProperties.sh, located at BPR_Home/bin, generate the keystore for RDU. This creates a server certificate keystore, which contains the private key and the associated client public key certificates.

```
./changeSSLProperties.sh -gk
```

Step 2 The following command generates, exports a self-signed certificate to a file and then exports it into the RDU certificate keystore.

```
./changeSSLProperties.sh -exp
```

Step 3 Set the private key password for RDU by running the following command:

```
./changeSSLProperties.sh -cpkp rdu
```

Step 4 Stop the RDU.

```
BPR_HOME/rdu/bin/stopRDU.sh
```

Step 5 Enable SSL on RDU by running the following command:

```
./changeSSLProperties.sh -ssl rdu enable
```

Step 6 Start the RDU.

```
BPR_HOME/rdu/bin/startRDU.sh
```

Step 7 Configure the clients to use the new server certificate keystore from the RDU.

Step 8 To ensure that the changes you make to the keystore take effect, you must restart the clients from the watchdog agent command line (see Using Prime Cable Provisioning Process Watchdog from CLI for the list of CLI commands).

Configuring SSL on DPE

This section describes how to configure SSL on the DPE services.

You can configure DPE security options from the DPE CLI. For more information, see the Cisco Prime Cable Provisioning 6.3 DPE CLI Reference Guide.

To enable SSL on DPE:

Step 1 To import the rootCA.crt file into cacerts trust store, navigate to BPR_HOME/bin and run the following command:

```
./changeSSLProperties.sh -imp /tmp/rootCA.crt rducert
```

Step 2 Run the command dpe rdu-server <host name> <port number> <secure> where the value for secure must be set to true.

For example:

```
dpe rdu-server bac-rhel5-vm80 49188 true
```

In the above command, you must enter the correct secure port number that RDU listens to. By default, RDU listens to the 49188 port.

Step 3 Restart the DPE by using the **dpe reload** command to ensure that the changes take effect.

Configuring SSL on API Client

You can configure SSL on any API client by using the api.properties file.

To enable SSL on an API client:

Step 1 Import the signed rootCA certificate into the JRE of the API client.

Step 2 In the api.properties file, for the property /server/rdu/secure/enabled, set the value to true.

For example:

```
/server/rdu/secure/enabled=true
```

Step 3 Provide the secure RDU server details using the property /rdu/secure/servers. The value for the property is a comma separated list of host and port number as shown below

For example:

```
/rdu/secure/servers=<hostname:port, hostname2:portnum2>
/rdu/secure/servers=bac-rhel5-vm80:49188,64.103.255.6:49188
```

Step 4 Restart the API client to ensure that the changes take effect.

Configuring SSL on Prime Network Registrar Extension Point

You can use the changeNRProperties.sh tool to set the secure communication on Prime Network Registrar Extension Point.

To enable SSL on CPNR-EP:

Step 1 Copy the rootCA.pem file to the BPR_HOME/lib/security folder.

Step 2 Run the following command:

```
./changeNRProperties.sh -ssl enable
```

Step 3 Set the secure port:

```
./changeNRProperties.sh -p 49188
```

Step 4 Restart the DHCP server.

Overriding PACE Connection Settings using api.properties

Prime Cable Provisioning allows you to override your existing PACE connection signatures for API clients without having to recompile your existing Java sources. This is achieved by defining additional properties in your api.properties. Properties such as port information and switching between secure and non-secure connections can be controlled via your api.properties.

To enable secure communication between an API client and it's RDU:

Step 1 Set /server/rdu/secure/enabled property to true. Before you enable the secure communication, ensure that you have configured your certificates properly.

For example:

```
/server/rdu/secure/enabled=true
```

Step 2 Define a property /rdu/secure/servers, with the RDU hostname and secure port details. In case your API client talks to multiple RDUs then define a comma separated list of all your secure host names and port details.

For example:

```
/rdu/secure/servers=<hostname1:port1>,<hostname2:port2>,<hostname3:port3>
```

If you change the default non-secure port number(49187) on the RDU, for the change to take effect, you need to set the property /rdu/unsecure/servers with the correct port information. You need not modify the Java sources.

For example:

```
/rdu/unsecure/servers=<hostname1:port1>,<hostname2:port2>,<hostname3:port3>, with your RDU
 hostnames and ports
```

At any given time, an API client can communicate to an RDU either in secure or non-secure mode but not both. If it attempts to communicate to the RDU in a mode different from the existing one, an exception is raised. In order to create a connection in the other mode, you must release all live PACE connections of the existing mode. However, the same API client can communicate with other RDUs in different modes.

Note If you have defined the same host name for both secure and non-secure communication, then the secure communication takes precedence over non-secure communication.

Signing a Certificate

While installing Prime Cable Provisioning, you can either self-sign the certificate of opt to get it signed from an external signing authority. In case of a self-signed certificate, the certificate encrypts your communication interactions. Since a self-signed certificate is not signed by a signing authority, web browsers flag the certificate as potentially risky. To avoid this you must import your signed certificate as a replacement.

Signing a Certificate Through an External Authority

This section explains how to get the certificate signed through an external authority.

Note To enable the use of client certificates for server authentication, ensure that the public certificate of the signing authority for server certificates is loaded into the cacerts keystore. Follow the procedure described in Importing Signing Authority Certificate into Cacerts Keystore, on page 11.

To sign a server certificate:

Step 1 Generate a new private key for the RDU using keytool in case you have not created it while installing the RDU server. See Generating Private Key for a New RDU Certificate, on page 9.

Step 2 Generate a certificate-signing request (CSR). See Generating a Certificate-Signing Request, on page 10.

Step 3 Request a public certificate from the signing authority by using CSR.

Step 4 Load the public key of the signing authority into the cacerts keystore. See Importing Signing Authority Certificate into Cacerts Keystore, on page 11.

Step 5 Load the signed server certificate into the server keystore. See Importing Signed Certificate into Server Certificate Keystore, on page 12.

Step 6 Restart the RDU by using the command, `BPR_HOME/agent/bin/bprAgent restart rdu`, from the watchdog agent command line (see Using Prime Cable Provisioning Process Watchdog from CLI).

Self-signing a Certificate

Use the changeSSLProperties.sh tool to create a self-signed certificate. For the exact usage of the tool, see Using changeSSLProperties.sh.

On RDU

Step 1 Create a RDU keypair using the -gk option. Keep note of the rducert alias certificate password.

Step 2 Export the self-signed certificates using -exp option.

Step 3 Set the private key password for the RDU using the -cpkp option. Make use of the certificate password used while creating certificate using -gk option.

Step 4 Stop the RDU by executing `BPR_HOME/rdu/bin/stopRDU.sh`.

Step 5 Enable the SSL mode for the RDU server using -ssl rdu enable option.

Step 6 Start the RDU by executing `BPR_HOME/rdu/bin/startRDU.sh`.

On DPE

Step 1 Copy the RDU rootCA.crt file that is being generated in the previous step and is located under the BPR_HOME/lib/security directory to a desired location in the DPE server.

Step 2 Use the -imp option to import the certificate to the truststore.

Step 3 Enable RDU SSL communication on DPE by running the command, *dpe rdu-server <hostname><port> true*.

Step 4 For the changes to take effect, reload the DPE or PWS server.

On PWS

Step 1 Copy the RDU rootCA.crt file that is being generated in the previous step and is located under the BPR_HOME/lib/security directory to a desired location in the PWS server.

Step 2 Use the -imp option to import the certificate to the truststore.

Step 3 Add the secure host details by executing *changeSSLproperties.sh -csp api*. This displays the host and port details.

Step 4 Add or modify the list with the correct host name and port details.

Step 5 For the changes to take effect, reload the PWS server.

On CNR-EP

Step 1 Copy the RDU rootCA.pem file that is being generated in the previous step and is located under the BPR_HOME/lib/security directory to the BPR_HOME/lib/security directory of the CNR-EP server.

Step 2 Enable RDU SSL communication on CNR-EP by running the command *changeNRProperties.sh -ssl enable*.

Step 3 Set the secure port by executing *changeNRProperties.sh -p 49188*.

Step 4 For the changes to take effect, reload the DHCP server by executing the command: *CNR_HOME/local/usrbin/nrcmd dhcp reload*.

Importing an Existing Signed Server Certificate

If you already have the signed server certificate and you want to load it into the keystore, you must know the private key that is associated with the certificate. In this case, instead of following the procedure described above, follow the steps outlined in this section. Use the PCKS#12 file format, which combines both the private key and the signed certificate. You can load this file into a keystore by using the **keystore import-pkcs12** command.

To configure a server certificate with an existing signed server certificate:

Step 1 Load the existing private key and certificates into a RDU-compatible file, used in authenticating the RDU to SSL clients, by using the **keystore import-pkcs12** command.

When using this command, the syntax is:

```
# ./keystore import-pkcs12 keystore-filename pkcs12-filename keystore-password key-password
export-password export-key-password
```

- *keystore-filename*—Identifies the keystore file to create. If it already exists, it will be overwritten.

  Note Remember to specify the full path of the keystore file.

- *pkcs12-filename*—Identifies the PKCS#12 file from which you intend to import the key and certificate.
- *keystore-password*—Identifies the private key password and the keystore password that you used when you created your keystore file. This password must be between 6 and 30 characters.
- *key-password*—Identifies the password used to access keys within RDU keystore. This password must be between 6 and 30 characters.
- *export-password*—Identifies the password used to decrypt the key in the PKCS#12 file. The export password must be between 6 and 30 characters.
- *export-key-password*—Identifies the password used to access keys within the PKCS#12 keystore. This password must be between 6 and 30 characters.

For example:

```
# ./keystore import-pkcs12 example.keystore example.pkcs12 changeme changeme changeme changeme
% Reading alias [1]

% Reading alias [1]: key with format [PKCS8] algorithm [RSA]

% Reading alias [1]: cert type [X.509]

% Created JKS keystore: example.keystore

% OK
```

Step 2 Copy the new keystore file into the RDU *BPR_HOME/dpe/conf* directory.

Step 3 At the CLI, configure one of the RDU services to use the new keystore. See Configuring SSL Post Installation, for details.

Step 4 Restart the RDU the command `BPR_HOME/agent/bin/bprAgent restart rdu` from the watchdog agent command line (see Using Prime Cable Provisioning Process Watchdog from CLI).

Using Keytool Commands

The keytool utility uses command arguments to configure the keystore. The following table lists the keytool commands and their descriptions.

Table 1: Keytool Commands

Keytool Commands	Description
-alias *alias*	Identifies the identity assigned to a keystore entry, which stores the certificate chain and the private key. Subsequent keytool commands must use the same alias to refer to the entity.
-dname *dname*	Identifies the X.500 Distinguished Names used to identify entities, such as those that the subject and the issuer named.
-file *csr_file*	Identifies the CSR file to be exported.
-file *cert_file*	Identifies the file from which the certificate is to be read.
-keyalg *keyalg*	Identifies the algorithm to be used for key-pair creation. The values are DSA (default) and RSA.
-keysize *keysize*	Specifies a keysize, whose values must be in multiples of 64 bits.
-keypass *keypass*	Identifies the password assigned to a key pair.
-keystore *keystore*	Customizes the name and location of a keystore.
-noprompt	Specifies that no prompts are to be issued during an import operation.
-provider *provider_class_name*	Identifies the name of the cryptographic service provider's master class file when the service provider is not listed in the security properties file.
-rfc	Specifies that the output of the MD5 fingerprint of a certificate, which appears in printable-encoding format.
-storepass *storepass*	Identifies the password assigned to a keystore.
-sigalg *sigalg*	Specifies the algorithm to be used to sign the certificate.

Keytool Commands	Description
-storetype *storetype*	Identifies the type assigned to a keystore or an entry into a keystore.
-trustcacerts	Specifies that additional certificates are considered for the chain of trust.
-v	Specifies that the output of the MD5 fingerprint of a certificate is printed in human-readable format.
-validity *valDays*	Identifies an expiration period. The default is 90 days.

Note For additional information on keytool and general certificate-management concepts, refer to Oracle documentation.

Generating Private Key for a New RDU Certificate

The **keytool -genkey** command generates a key pair (a public key and an associated private key), and wraps the public key into an X.509 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are stored in a new keystore entry identified by alias.

Note The purpose of an *alias* is to uniquely identify a key pair within the keystore, in case you have multiple key pairs. In the context of Prime Cable Provisioning, the *alias* used for the RDU key is critical. Prime Cable Provisioning uses rducert as its default *alias*. In case you have changed the *alias*, add or update the /secure/rdu/certificateAlias property with the new *alias* in the rdu.properties file.

Note If you are directly using the keytool to generate the RDU keypair, you must use the *changeSSLProperties.sh -cpkp* command to update the private key password that you provided while creating the keypair to the rdu.properties file.

The following example uses *.keystore* as the name of the keystore file.

```
# ./keytool -genkey -alias rducert -storetype JCEKS -validity 730 -keyalg RSA -keystore
/opt/CSCObac/lib/security/.keystore

Enter keystore password: changeit
Re-enter new password: changeit
What is your first and last name?
  [Unknown]:  BAC Testing
What is the name of your organizational unit?
  [Unknown]:  NMTG
What is the name of your organization?
  [Unknown]:  Cisco Systems Inc.
What is the name of your City or Locality?
  [Unknown]:  Bangalore
What is the name of your State or Province?
```

```
  [Unknown]:  KAR
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN correct?
  [no]:  yes
Enter key password for <rducert>
        (RETURN if same as keystore password):
```

Displaying Self-Signed Certificate

The **keytool -list** argument displays the contents of the keystore entry identified by alias. If you do not specify an alias, the entire contents of the keystore appear.

If you combine **-list** with **-v**, the certificate chain associated with the alias appears. The following **keytool -list** sample output displays the keystore containing a single self-signed certificate.

```
# ./keytool -list -v -storetype JCEKS -keystore /opt/CSCObac/lib/security/.keystore
Enter keystore password: changeit
Keystore type: JCEKS
Keystore provider: SunJCE
Your keystore contains 1 entry
Alias name: rducert
Creation date: Aug 21, 2012
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Issuer: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Serial number: 50331e4f
Valid from: Tue Aug 21 11:06:15 IST 2012 until: Mon Nov 19 11:06:15 IST 2012
Certificate fingerprints:
         MD5:  8F:03:43:33:51:9B:DB:C5:0E:27:B5:B4:A1:FE:97:83
         SHA1: A1:FB:63:CD:58:44:A0:CA:1A:A2:41:9B:09:C1:CC:5E:EB:66:B9:96
         Signature algorithm name: SHA1withRSA
         Version: 3
```

Generating a Certificate-Signing Request

At this point in the procedure, the keystore contains a private key and a X.509 self-signed certificate. If the RDU tries to respond with this certificate to a client's initial handshake, the client will reject the certificate with a TLS alert bad CA, indicating that the certificate authority that the client trusted did not sign the certificate. Therefore, the signing authority that the client trusts must sign the certificate.

Note To support SSL, the clients must have a list of preconfigured public certificates of signing authorities that they trust.

The **keytool -certreq** command parameter generates a certificate-signing request (CSR). This command generates the CSR in the industry standard PKCS#10 format.

The following example uses a keystore with a pre-existing self-signed certificate under **alias** *rducert* to generate a certificate-signing request and output the request into the train-1.csr file.

```
# ./keytool -alias rducert -certreq -file /opt/CSCObac/lib/security/rducert.csr -storetype
```

```
 JCEKS -keystore /opt/CSCObac/lib/security/.keystore
Enter keystore password: changeit
```

The next step is to submit the CSR file to your signing authority. Your signing authority or your administrator, who is in possession of the private key for the signing authority, will generate a signed certificate based on this request. From the administrator, you must also obtain the public certificate of the signing authority.

Verifying the Signed Certificate

After you have received the signed certificate, use the **keytool -printcert** command to verify if the self-signed certificate is in the correct file format and uses the correct owner and issuer fields. The command reads the certificate from the **-file** *cert_file* parameter, and prints its contents in a human-readable format.

The *rootCA.crt* file in this example identifies the signed certificate that the administrator provides.

```
# ./keytool -printcert -file rootCA.crt
Owner: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Issuer: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Serial number: 50331e4f
Valid from: Tue Aug 21 11:06:15 IST 2012 until: Mon Nov 19 11:06:15 IST 2012
Certificate fingerprints:
         MD5:  8F:03:43:33:51:9B:DB:C5:0E:27:B5:B4:A1:FE:97:83
         SHA1: A1:FB:63:CD:58:44:A0:CA:1A:A2:41:9B:09:C1:CC:5E:EB:66:B9:96
         Signature algorithm name: SHA1withRSA
         Version: 3
```

Note The keytool can print X.509 v1, v2, and v3 certificates, and PKCS#7-formatted certificate chains comprising certificates of that type. The data to be printed must be provided in binary-encoding format, or in printable-encoding format (also known as Base64 encoding) as defined by the RFC 1421.

Importing Signing Authority Certificate into Cacerts Keystore

Before importing the certificate into the server certificate keystore, you must import the public certificate of the signing authority into the cacerts keystore; because when a certificate is being imported into the keystore, the keytool checks if a chain of trust can be established between the certificate and its signing authority. If a chain of trust cannot be established, an error message appears.

Note The cacerts file bundled with Prime Cable Provisioning ships with several root certificate common third-party signing authorities. You can manage the cacerts keystore by using the keytool utility. The default cacerts keystore password is **changeit**. The cacerts database file resides in the *BPR_HOME/jre/lib/security* directory.

The cacerts keystore does not need to be copied anywhere. The client will use the new keystore as soon as it is restarted.

```
# ./keytool -import -alias rducert-file rootCA.crt -keystore
/opt/CSCObac/jre/lib/security/cacerts
Enter keystore password:  changeit
Owner: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Issuer: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Serial number: 50331e4f
Valid from: Tue Aug 21 11:06:15 IST 2012 until: Mon Nov 19 11:06:15 IST 2012
```

```
Certificate fingerprints:
         MD5:  8F:03:43:33:51:9B:DB:C5:0E:27:B5:B4:A1:FE:97:83
         SHA1: A1:FB:63:CD:58:44:A0:CA:1A:A2:41:9B:09:C1:CC:5E:EB:66:B9:96
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Note The keytool can import X.509 v1, v2, and v3 certificates, and PKCS#7-formatted certificate chains comprising certificates of that type. The data to be imported must be provided in binary-encoding format, or in printable-encoding format (also known as Base64 encoding) as defined by the RFC 1421.

Importing Signed Certificate into Server Certificate Keystore

Once you import the public certificate of the signing authority into the cacerts keystore, you must import the signed server certificate into the RDU server certificate keystore. You will already have a keystore with private key and corresponding self-signed certificate (public key).

By importing the certificate reply (signed certificate), the keystore is modified to associate the signed certificate with the existing private key in the server certificate keystore.

When importing the certificate reply into the keystore, you must use the **-trustcacerts** flag with the **-import** command for certificates in the *cacerts* file to be used to establish chains of trust with the certificate reply in the subject's keystore.

Keytool -import (Signed Server Certificate)

```
# ./keytool -import -trustcacerts -file rducert.crt -keystore
/opt/CSCObac/lib/security/.keystore -alias rducert -storetype JCEKS
Enter key password: changeme
Enter keystore password:  changeme
Certificate reply was installed in keystore
Certificate was added to keystore
```

After you import the signed server certificate into the RDU server certificate keystore, use the keytool **-printcert** command to verify the keystore contents, as outlined in Verifying the Signed Certificate, on page 11. The **-printcert** output should now show the issuer to be the signing certificate authority, and that a chain of trust has been established using the signing authority with the root trusted certificate.

Troubleshooting SSL

Prime Cable Provisioning supports SSL logging for RDU, DPE, CPNR extensions and API clients. This helps in handling connection issues during SSL communication.

The SSL communication logs are updated in the following log files:

- rdu.log - You must enable debug logging with secure messaging on for SSL log to appear in rdu.log.
- dpe.log - You must enable secure messaging via DPE CLI to enable DPE SSL logging.
- name_dhcp_1_log - You must enable trace level 4 for the DHCP process for debugging DHCP SSL issues.

The client API log can be configured and set to any name based on the configuration.

Ciphers Supported for Secure Communication

The default cipher list contains the following ciphers, which contains strong ciphers for TLS 1.2 when used in Secure mode.

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
- TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
- TLS_DHE_DSS_WITH_AES_128_GCM_SHA256