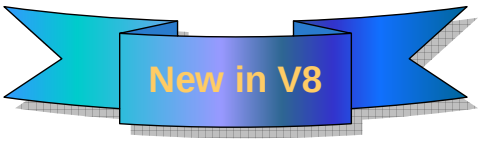
MQ SSL/TLS Channels Including V8 changes

Morag Hughson

hughson@uk.ibm.com

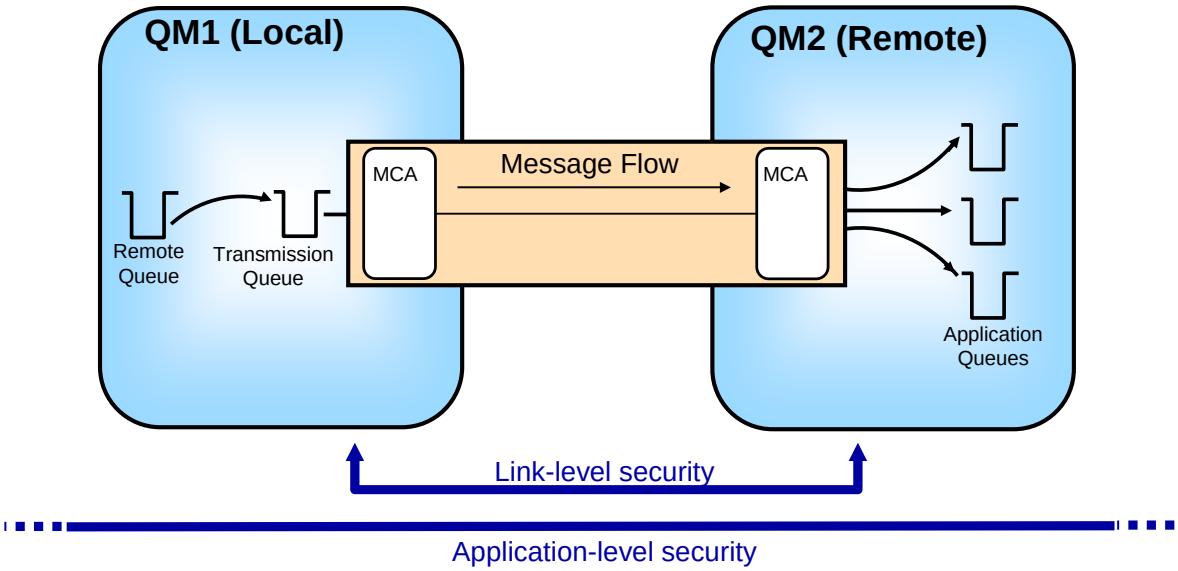
Agenda

- **MQ Configuration Tasks**
  - With some minor V8 updates
- **Major new feature in MQ V8**



- **Associating a certificate with an MQ entity**
  - Queue Manager
  - WebSphere MQ client
- **Certificate Revocation**
- **New channel attributes**
  - Specifying CipherSpec
  - Specifying permitted partners
  - Specifying that the partner must provide a certificate
- **Using Secret Key Reset**
- **Using Channel Status**
- **Refreshing WebSphere MQ**
  - Certificate replacement
- **Miscellaneous**
  - Specifying certified cryptography should be used
  - Specifying cryptographic hardware (some platforms)
  - Specifying SSL tasks (z/OS)

WebSphere MQ and SSL



WebSphere MQ and SSL - Notes

N

O

T

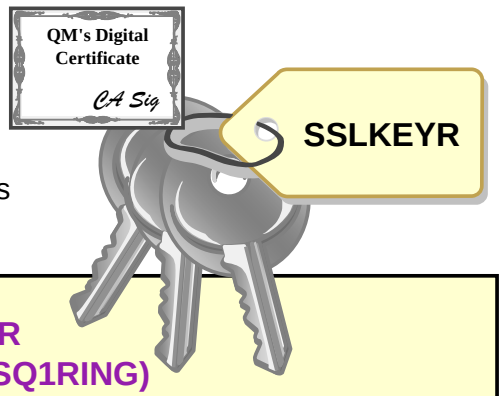
E

S

- We can look at the security of MQ at two levels which I will describe as application level security and link level security.
- SSL is used in WebSphere MQ to do channel authentication and link-level encryption. This means that your messages are encrypted when they are moved across the network but they are not encrypted when they reside on your queues. If your security set-up is such that your queues are secure, then link-level security may be all that you require.
- If your system already authenticates applications and encrypts the messages at MQPUT time and decrypts them at MQGET time, then the messages are encrypted as they reside on the queues. They should not require further encryption as they are transported across the network. In this scenario, link-level security may not be required, or may only be required for queue manager to queue manager authentication at a minimum.

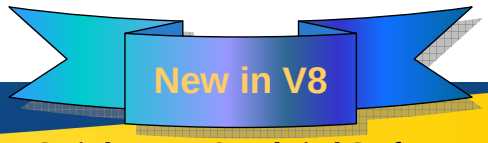
Queue Manager Certificate

- **Key Repository**
  - Contains
    - Queue Manager's own Digital Certificate
    - Digital Certificates from various Certification Authorities
  - On Unix®, Windows®, iSeries® QMgrs
    - Key database path
  - On z/OS® Queue Managers
    - Keyring name
- **Default label**
  - z/OS Queue Manager
    - ibmWebSphereMQ<QMgr Name> (mixed case) label
  - Distributed Queue Manager
    - ibmwebspheremq<qmgr name> (lower case) label
- **In V8 can use the QMGR CERTLABL attribute**




```
ALTER QMGR
SSLKEYR(CSQ1RING)
CERTLABL('CSQ1Certificate')
CERTQSGL('SharedCert')

ALTER QMGR
SSLKEYR('var/mqm/qmgrs/QM1/ssl/key')
CERTLABL('QM1Certificate')
```



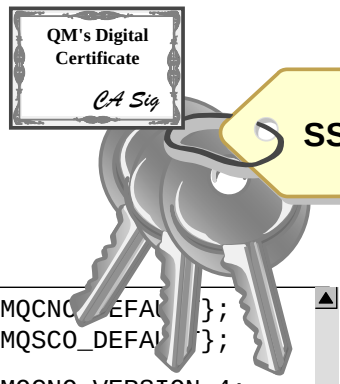
Queue Manager Certificate – Notes

N O T E S

- A digital certificate contains the identity of the owner of that certificate. Each WebSphere MQ queue manager has its own certificate. On all platforms this certificate is stored in a key repository using your digital certificate management tool, e.g. in RACF® (z/OS) or iKeyMan (UNIX and Windows).
- The key repository is specified on the WebSphere MQ QMGR object using the ALTER QMGR command. On z/OS this is the name of the keyring object in the External Security Manager (ESM), and on the distributed platforms this is the path and the stem of the filename for the key database file.
- The key repository generally also contains a number of signed digital certificates from various Certification Authorities which allows it to be used to verify certificates it receives from its partner at the remote end of the connection.
- Before WebSphere MQ V8, the label name for a digital certificate to be used by the queue manager was fixed by MQ. You had to label your certificate exactly as WebSphere MQ required it, in order for the certificate to be found. This doesn't always meet customer standards of certificate labelling.
- On z/OS, the unless told otherwise, MQ looks for a certificate in the key repository labeled as ibmWebSphereMQ<QMgr Name>. On UNIX, Windows and iSeries, the certificate label searched for is the lower-case label ibmwebspheremq<qmgr name> . Note that the certificate label is also sometimes referred to as its "friendly name".
- In WebSphere MQ V8 you can provide your own label name for the queue manager to use with a new attribute on ALTER QMGR called CERTLABL (and additionally CERTQSGL on z/OS for a QSG level certificate – previously located with the label ibmWebSphereMQ<QSG-name>).

Client Certificate

- **Key Repository**
  - Contains
    - Queue Manager's own Digital Certificate
    - Digital Certificates from various Certification Authorities
  - mqclient.ini file
    - SSL Stanza – SSLKeyRepository
  - MQCONNX (MQSCO structure)
    - KeyRepository
  - Environment variable
    - export MQSSLKEYR=var/mqm/ssl/key
- **Default Label**
  - ibmwebspheremq<logged on userid> (lower case) label
- **In V8 can name Client Certificate**
  - mqclient.ini file SSL Stanza
    - CertificateLabel
  - MQCONNX (MQSCO structure)
    - CertificateLabel
  - export MQCERTLABL=MyCert




```
MQCNO cno   = {MQCNO_DEFAULT};
MQSCO sco   = {MQSCO_DEFAULT};

cno.Version =  MQCNO_VERSION_4;
sco.Version =  MQSCO_VERSION_5;
memcpy(sco.KeyRepository, ... );
memcpy(sco.CertificateLabel,..);
cno.SSLConfigPtr = &sco;
MQCONNX(QMName,
        &cno,
        &hConn,
        &CompCode,
        &Reason)
```

New in V8

mqclient.ini
SSL:
SSLKeyRepository=C:\key
CertificateLabel=MyCert

Client Certificate – Notes

N O T E S

- A digital certificate contains the identity of the owner of that certificate. Generally each user of the WebSphere MQ client has a separate key repository file, with access restricted to that user.
- This key repository file is accessed using the environment variable MQSSLKEYR, or the MQCONNX SSLKeyRepository parameter.
- The key repository generally also contains a number of signed digital certificates from various Certification Authorities which allows it to be used to verify certificates it receives from its partner at the remote end of the connection.
- Before WebSphere MQ V8, the label name for a digital certificate to be used by an MQ Client was fixed by MQ. You had to label your certificate exactly as WebSphere MQ required it, in order for the certificate to be found. This doesn't always meet customer standards of certificate labelling.
- Unless told otherwise the MQ client code uses the certificate labeled with ibmwebspheremq followed by the logon userid, wrapped to lower case.
- In WebSphere MQ V8 you can provide your own label name for the MQ Client to use.
- For clients, you can provide the Certificate label in the MQSCO structure (along with the SSLKeyRepository location); or in the SSL stanza in the mqclient.ini file (along with the SSLKeyRepository location), or using the environment variable MQCERTLABL.

Certificate Revocation

- **Define AUTHINFO objects**

```
DEFINE AUTHINFO(LDAP1)
       AUTHTYPE(CRLLDAP)
       CONNAME('ldap(389)')
       LDAPUSER('cn=user')
       LDAPPWD(...)

DEFINE AUTHINFO(OCSP1)
       AUTHTYPE(OCSP)
       OCSPURL('http://ocsp.server')
```

LDAP Conname
LDAP UserName
LDAP Password

LDAP Server

OCSP Server

- **Put these AUTHINFO objects into a namelist**
- **Associate namelist with QMGR**
  - SSLCRLNL attribute
- **Clients: Client Channel Table or MQCONNX**
  - Client Channel Table contains the AUTHINFO definitions present when table copied off

Certificate Revocation - Notes

N O T E S

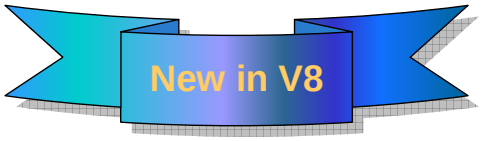
- **Queue Manager**
  - Certificate Revocation Lists (CRLs) and Authority Revocation Lists (ARLs) can be stored in and accessed from Lightweight Directory Access Protocol (LDAP) servers. A few parameters must be specified to be able to access an LDAP server containing CRLs and ARLs. These parameters are the DNS name or IP address of the LDAP Server with an optional TCP/IP port number; also optionally the Distinguished Name of the entry that is binding to the directory (e.g. C=UK, O=IBM, OU=Development, CN=John Smith), and the password associated with the Distinguished Name. Note that LDAP CRL/ARL servers are generally defined to be publicly readable.
  - These parameters are defined on a queue manager object, an AUTHINFO object. Several of these objects may be needed to ensure redundancy so that, for example, if the first LDAP server connected to is down, another can be connected to that will be able to supply the same information. Up to ten of these AUTHINFO objects can be supplied for CRL/ARL checking. The list of AUTHINFO objects is named in a namelist and this namelist is specified on the SSLCRLNL queue manager attribute using the ALTER QMGR command.
  - The above mechanism for accessing LDAP CRLs is not available on OS/400. On OS/400, LDAP CRLs are accessed using Digital Certificate Manager (DCM).
  - An alternative to CRLs on LDAP servers (available on Windows and Unix) are the use of OCSP servers. The URL to connect to the OCSP server can either be used directly from the certificate where it may be included by the CA as a certificate extension, or configured in an AUTHINFO object.
- **Clients**
  - When the channel runs, the client channel definition table AUTHINFO information does not have to match the definitions current on the queue manager system at that stage.
  - MQCONNX provides a structure, MQAIR (MQ auth info records), to allow AUTHINFO details to be specified.

SSLCIPH

- **Only mandatory parameter on an SSL channel**
  - Without it channel is assumed not to be using SSL
  - Specify the CipherSpec to be used
  - Both ends must specify the same CipherSpec

ALTER CHL … SSLCIPH(TLS_RSA_WITH_NULL_SHA256)
or
SSLCIPH(003B)

- **From a list of human-readable strings**
  - z/OS and IBM i: also SSL API numeric values
    - Allow support of new CipherSpecs without updates to MQ Code
  - Full list in Knowledge Center
    http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.sec.doc/q014260_.htm
- **SHA-2 algorithms available everywhere in V8**
  - And earlier by APAR, see technote
    http://www.ibm.com/support/docview.wss?uid=swg21639606



Platform support	CipherSpec name	Protocol Used	Data Integrity	Encryption Algorithm	Encryption Bits	FIPS	Suite B
IBMi Linux Windows UNIX z/OS		SSL 3.0 TLS 1.0 TLS 1.2	MD5 SHA-1/2* AEAD*	RC2/4 [3]DES AES	40, 56 128, 168 256	Yes / No	128/192 bit No

SSLPEER

- **Specify the partner's Distinguished Name**
- **Can use wildcards**
- **Multiple Organisational Unit (OU)**
  - Must be specified in descending hierarchical order (e.g. OU=Big Unit, OU=Medium Sized Unit, OU=Small Unit)
- **Can get much more flexibility using CHLAUTH rules for peer name matching**
  - More on this later

SSLPEER('CN="Morag Hughson", O=IBM')

or

SSLPEER('OU=WebSphere*, O=IBM')

SSLCAUTH

- **Client Authentication**
  - Request whether the client end is required to provide a certificate
  - for authentication

  - N.B. Client refers to SSL Client, i.e. initiating end of session

SSLCAUTH(REQUIRED)

or

SSLCAUTH(OPTIONAL)

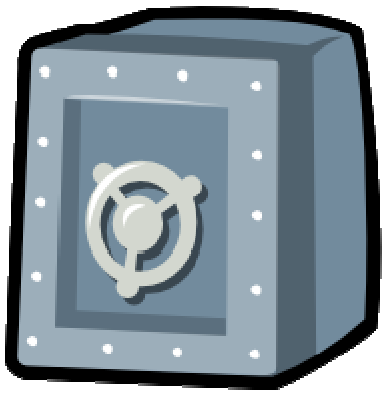
New Channel Attributes - Notes

N O T E S

- There are three channel attributes that can be used when setting up SSL on your TCP/IP channels.
- **SSLCIPH**
  - This is the channel parameter that you use to specify the CipherSpec to be used by the channel. The same CipherSpec must be specified at both ends of the channel for the SSL session to be successfully established. The values used are most often the string values listed in the Knowledge Center which are a human-readable string combining the encryption algorithm and the hash function to be used. The corresponding numeric values for the O/S SSL API can also be used on z/OS and iSeries, thus allowing new CipherSpecs to be supported without updates to WebSphere MQ code. Different types of input can be successfully used on the two ends of a channel as long as they specify the same CipherSpec. O/S API values are not relevant on UNIX and Windows as on these platforms SSL support (GSKit) is part of Websphere MQ. For a list of those numbers see on z/OS:- http://www.ibm.com/support/knowledgecenter/api/content/SSFKSJ_8.0.0/com.ibm.mq.ref.doc/csq_x.htm#csq_x__csqx631e
- **SSLPEER**
  - This parameter is used to check against the Distinguished Name from the partner's certificate. This field can have wildcards to allow generic matching. If the field is left blank or is not present then no checking is done against the partner's Distinguished Name within the WebSphere MQ code. Generally speaking we now advise to use CHLAUTH for peer name matching as it is much more flexible - more on that later.
- **SSLCAUTH**
  - The end of the channel which initiates the SSL connection is considered by SSL to be the client. The client always authenticates the server's certificate, but may not necessarily send a certificate to the server to be authenticated. This parameter is used on the SSL server end of the connection to say whether we expect a certificate for authentication from the SSL client, or whether only the server end will have its certificate authenticated. This may be useful for WebSphere MQ client connection, or for lightweight queue managers, or indeed for any initiating partner where authentication is not deemed necessary. This parameter can have the value OPTIONAL or REQUIRED. The default is REQUIRED.

Using Secret Key Reset

- **Periodic renegotiation**
  - After a specified number of bytes of data has flowed
  - Before sending data after an idle period in which heartbeat(s) have flowed
- **Set specified number of bytes**
  - SSLRKEYC queue manager attribute
- **Display Channel Status shows**
  - The number of times the key has been reset
  - The time/date of the last reset



```
ALTER QMGR SSLRKEYC(999 999 999)

DISPLAY  CHSTATUS(*)
         SSLRKEYS
         SSLKEYDA SSLKEYTI
```


Using Secret Key Reset - Notes

N O T E S

- Secret Key Reset is a feature introduced in WebSphere MQ V6.
- An SSL channel will periodically renegotiate its secret key algorithm if Secret Key Reset is enabled. This renegotiation will take place after the specified amount of data has been moved across the channel, and also before data is next sent across a channel that has been idle for a period of time and has sent heartbeat flow(s).
- Fields in the channel status display will show when the last reset was done and how many resets have been done for the life of this channel instance.

Using Channel Status

- **Fields showing Partner Certificate details**
  - SSLPEER
    - Partner's Certificate Subject's Distinguished Name
  - SSLCERTI
    - Partner's Certificate Issuer's Distinguished Name
- **Mapped User ID (z/OS only)**
  - SSLCERTU
- **Also passed to Security exit and CHLAUTH rules**
  - More on CHLAUTH later

```
SSLPEER(CN=MQ23,T=Queue Manager,O=IBM,L=Hursley,ST=Hampshire,C=UK)
SSLCERTI(CN=WebSphere MQ CA,O=IBM,L=Hursley,ST=Hampshire,C=UK)
```

```
SSLPEER(CN=MQ23,T=Queue Manager,O=IBM,L=Hursley,ST=Hampshire,C=UK)
SSLCERTI(CN=MQ23,T=Queue Manager,O=IBM,L=Hursley,ST=Hampshire,C=UK)
```


Using Channel Status - Notes

N O T E S

- There are a number of fields in the channel status displays that are related to SSL channels.
- **SSLPEER and SSLCERTI**
- These two fields show the details of the partner's certificate. SSLPEER on DISPLAY CHSTATUS should not be confused with SSLPEER on DISPLAY CHANNEL. On the channel definition SSLPEER contains the filter to match against the partner's DN, and on channel status it shows the actual DN of the partner's certificate. SSLCERTI contains the issuer's DN from the partner's certificate. If SSLPEER and SSLCERTI are the same, the certificate is self-signed. These fields are also passed to the Security Exit so more complicated decisions can be made based on this information.
- **SSLCERTU**
- On z/OS, when mapping a certificate in RACF to a user ID, this field shows the user ID found.

Refreshing SSL on WebSphere MQ

- **Cached view(s) of Key Repository**
- **Changes require cached view(s) to be refreshed**
  - New certificates
  - Deleted certificates
  - Updated certificates
    - Replacing expiring certificates

REFRESH SECURITY TYPE(SSL)

- **This command**
  - Stops all SSL channels
  - New cached view(s) of the Key Repository created
  - Starts all SSL channels again
    - Sender types
    - Receiver types will restart when partner retries
  - Also picks up other changes
    - New Key repository location
    - New LDAP CRL/ARL locations

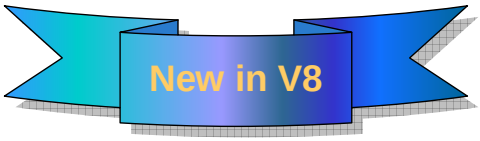
Refreshing SSL on WebSphere MQ - Notes

N O T E S

- This feature was introduced in WebSphere MQ V6.
- The SSL environment set up to run SSL channels in a channel process has a cached view of the key repository made at initialization time. If you make changes to your key repository, i.e. add, remove or update certificates, for example, because your are replacing a certificate that is about to expire, this cached view needs to be refreshed in order for the SSL channels to start using the new certificates.
- In order to refresh this cached view of the SSL environment, without disrupting any non-SSL channels, use the REFRESH SECURITY TYPE(SSL) command. This will stop all the SSL channels on the queue manager, new cached view(s) of the key repository will be made and all the sending type channels will be started again. Receiving type channels will get restarted as the partner end retries the connection.
- Non-SSL channels will be unaffected by this command and will continue to run.
- Also use this command to pick up other changes, such as a new Key Repository locations, or new LDAP CRL/ARL locations.

Using Cert Labels to replace expiring certs

- **Used to have to issue GSKit/RACF commands to rename certificate**
  - ibmwebspheremqqm1 -> ibmwebspheremqqm1old
  - ibmwebspheremqqm1new -> ibmwebspheremqqm1
  - REFRESH SECURITY TYPE(SSL)
- **Now just MQ commands when the time comes**
  - Current label is ‘QM1 Cert 2013’
  - ALTER QMGR CERTLABL(‘QM1 Cert 2014’)
  - REFRESH SECURITY TYPE(SSL)



Using Cert Labels to replace expiring certs - Notes

N O T E S

- It is worth highlighting here that the change over from using one certificate to another is now a task that can be accomplished by the MQ administrator alone, when he is ready. The job of installing the new certificate can be done at any prior point and labelled however you wish. That label does not now have to change in order to get the queue manager to use it, so it is just a task for the MQ administrator to tell the queue manager which label to use now, and then refresh.

Miscellaneous

- **Using only officially certified cryptography on an SSL**
  - Important for USA (and other) government bids
  - Can set up so can only use certain CipherSpecs
  - Federal Information Processing Standards (FIPS)
    - Added at WebSphere MQ V6.0 on Distributed platforms and V7.1 on z/OS
  - Suite-B
    - Added at WebSphere MQ V7.1 on Distributed platforms
- **CryptoGraphic Hardware on Unix and Windows**
  - Parameters are required by the SSL support.
  - requirements vary according to hardware used
  - Specify `ALTER QMGR SSLCRYP(<string>)`
    - <string>: cryptographic hardware configuration parameters
  - Clients: Environment variable or MQCONNX
    - SET MQSSLCRYP=<string>
    - SSLCryptoHardware
- **SSL Server MVS Tasks**
  - `ALTER QMGR SSLTASKS(8)`
    - Minimum 2 required to run SSL handshake and encryption calls on z/OS

Miscellaneous - Notes

N O T E S

Federal Information Processing Standards (FIPS)

- The US National Institute of Standards and Technology (NIST) is responsible for FIPS. The particular area of FIPS we are concerned with is the Cryptomodule Validation Program (CMVP), involving tests on individual cryptographic modules. At WebSphere MQ V6.0, on all Windows and Unix platforms some of the cryptographic modules provided within WebSphere MQ for use on SSL channels have been FIPS 140-2 certified. Only some CipherSpecs are acceptable for FIPS certification. It is possible to configure WebSphere MQ queue managers and clients to only accept CipherSpecs for which the WebSphere MQ cryptography has been FIPS certified. (Note that this does not guarantee FIPS certification if cryptographic hardware is used to provide the cryptography).

CryptoGraphic Hardware on some platforms

- SSLCryptoHardware is in the MQCONNX structure, MQSCO -- SSL Configuration Options.
- On other queue manager platforms the crypto-hardware is either configured automatically by the SSL software or is configured by the systems administrator independently of WebSphere MQ

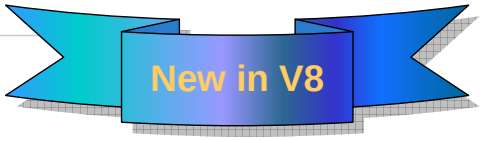
SSL Server MVS Tasks

- Server Tasks, similar to the adapter tasks already in the Channel Initiator address space, are required to run the SSL Handshake on z/OS. The number of these tasks started is specified using ALTER QMGR SSLTASKS. If there are not at least two of these tasks then no SSL channels will be able to start.

Request for Enhancement (26672)

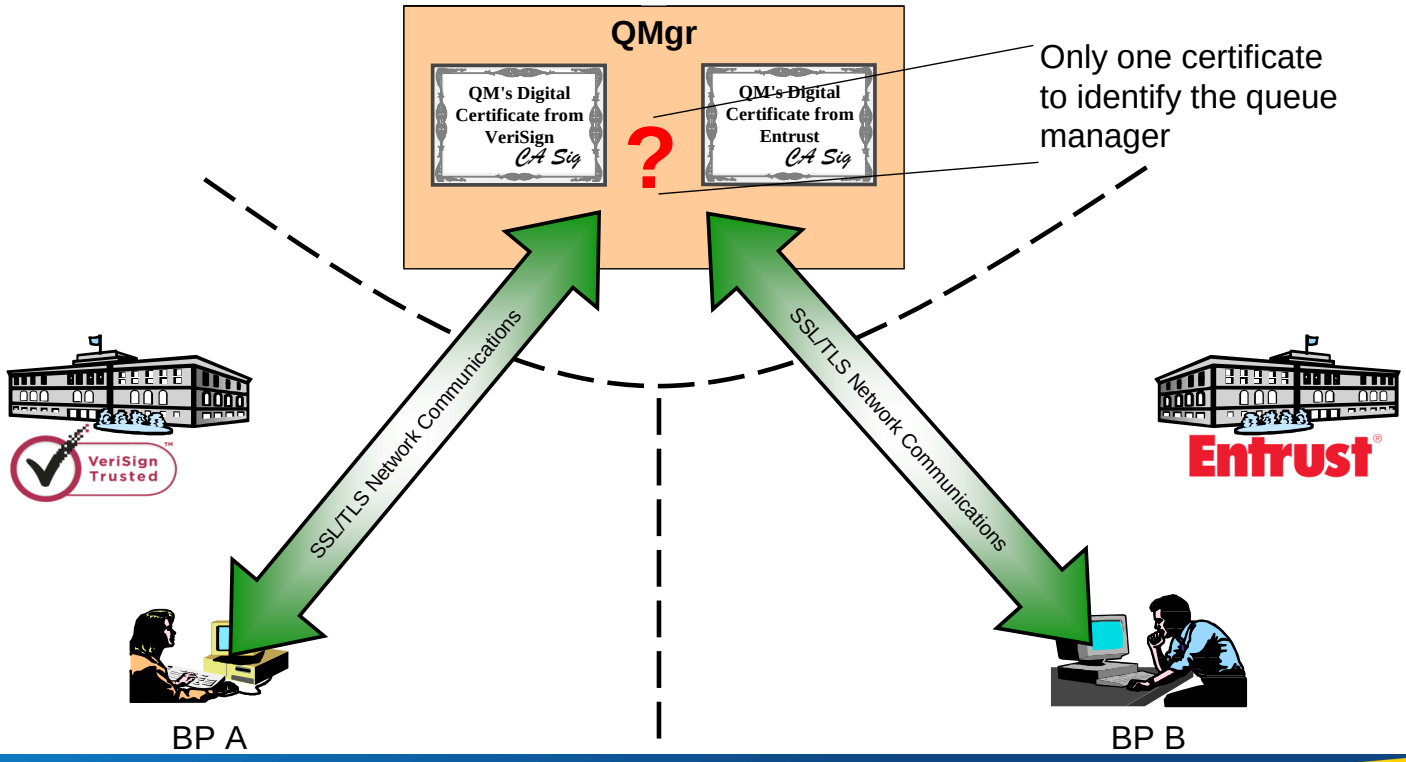
Headline:	Requesting the enhancement to support for SSL certificate per channel or group of channels
ID:	26672

Details | Comments | Attachments | Reconsideration | Release plans



Status:	Under Consideration
Visibility:	Public
Description:	Currently mq supports only one default signed certificate per queue manager. When one firm is connecting with multiple external firms, then any of these external firms can pretend to be a different external firm, if they can guess the channel name and sslpeername and connect. Especailly if the channel names and sslpeers are following certain naming conventions. Another problme is, every time when the certificate chain changes, every party that is connecting to this qmgr needs to refresh their store with the new chain. So having a certficate per channel or group channels instead of one certificate for all channels on the queue manager is the solution here. We would like IBM to consider this as high priority.
Use case:	The description itself is covering the use case scenario.
Bookmarkable URL:	http://www.ibm.com/developerworks/rfe/execute?use_case=viewRfe&CR_ID=26672 A unique URL that you can bookmark and share with others.

Business Partners with different CA requirements

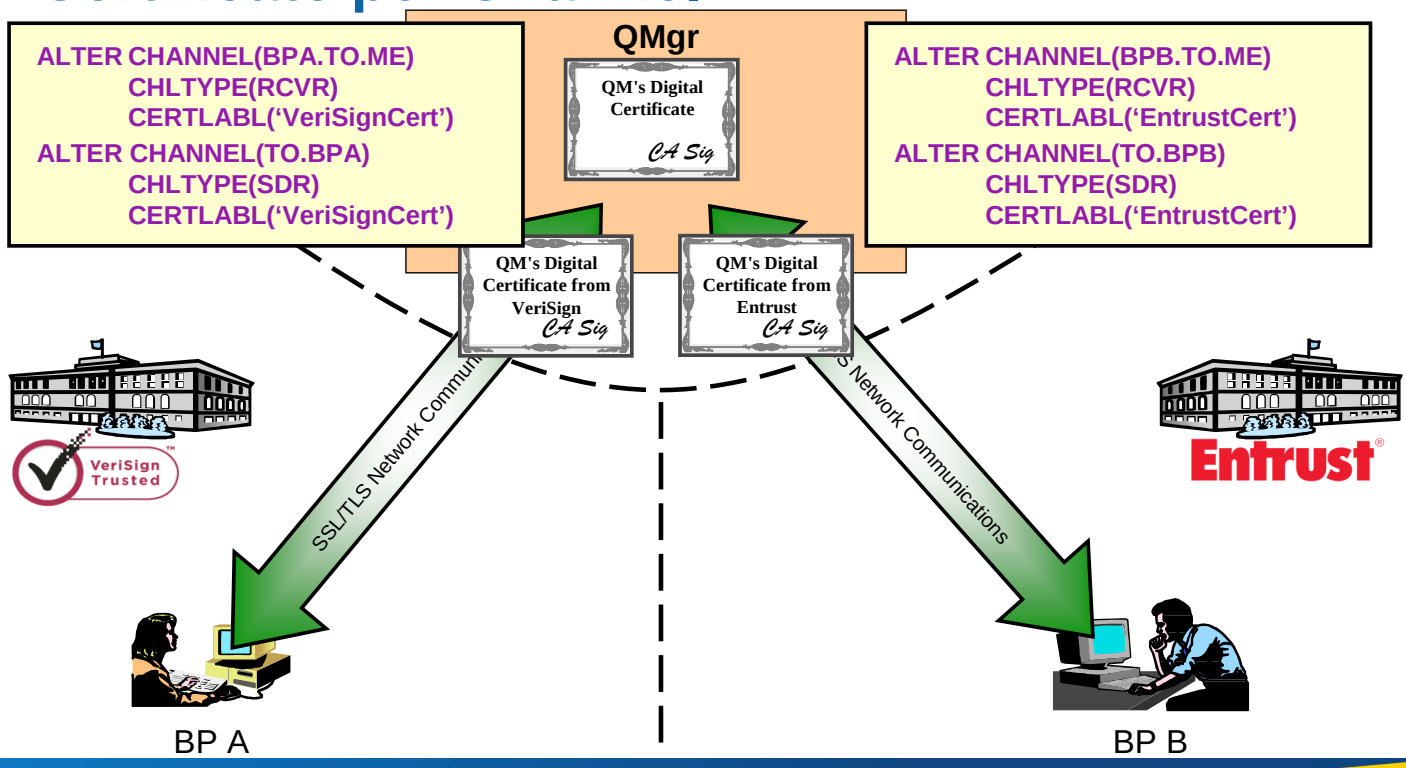


Business Partners with different CA requirements – Notes

N O T E S

- Imagine the situation where your company has need to communicate securely with two difference business partners. These business partners each have a different requirement about the Certificate Authority (CA) who signs the certificates that they are happy to accept. In our example, Business Partner A will only accept certificates signed by VeriSign, whereas Business Partner B will only accept certificates signed by Entrust.
- In order for your company to be able to communicate with both of these Business Partners, you need a certificate that is signed by VeriSign (to communicate with Business Partner A) and a certificate that is signed by Entrust (to communicate with Business Partner B). However, since a queue manager can only have one certificate, with releases prior to V8 of WebSphere MQ, you were forced into having two queue managers, one using each certificate. This is less than ideal.
- N.B. Some people also solve this issue by using an MQIPT in front of the queue manager.

Certificate per Channel

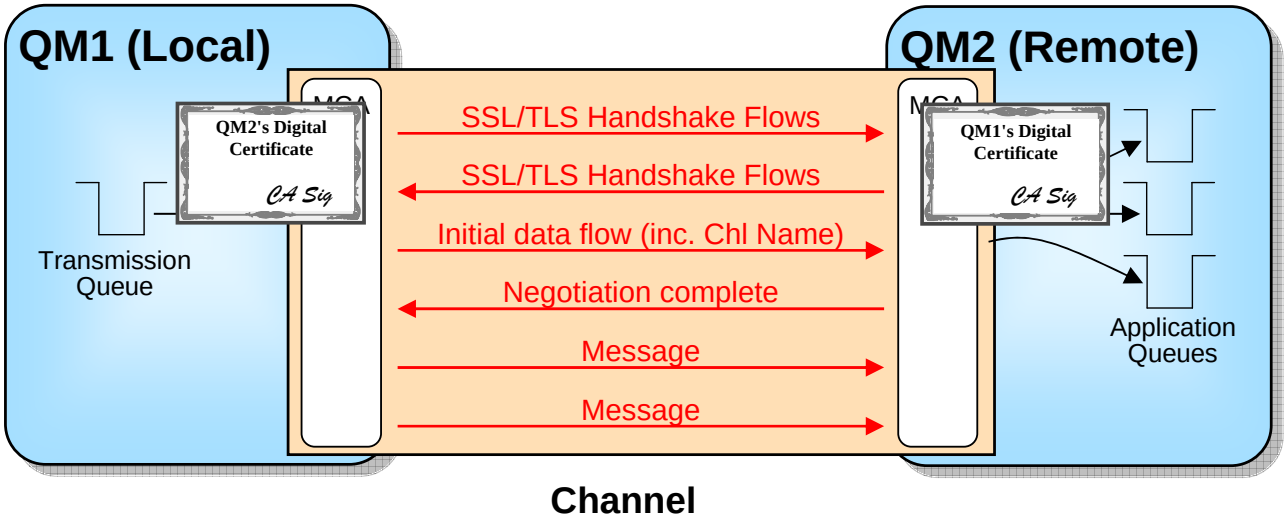


Certificate per Channel – Notes

N O T E S

- What is required is the ability to indicate that this particular channel should use a different certificate than other channels.
- This is achieved in WebSphere MQ V8 with an attribute on a channel, CERTLABL, which can either be blank – which means use whatever the queue manager overall is configured to use, or if provided, means that this channel should use the specifically named certificate.
- For reasons explained a little later on, we only allow you to specify a non blank CERTLABL at definition time if you are using a TLS cipherspec.

Why haven't we always done this?



Why haven't we always done this? – Notes

N O T E S

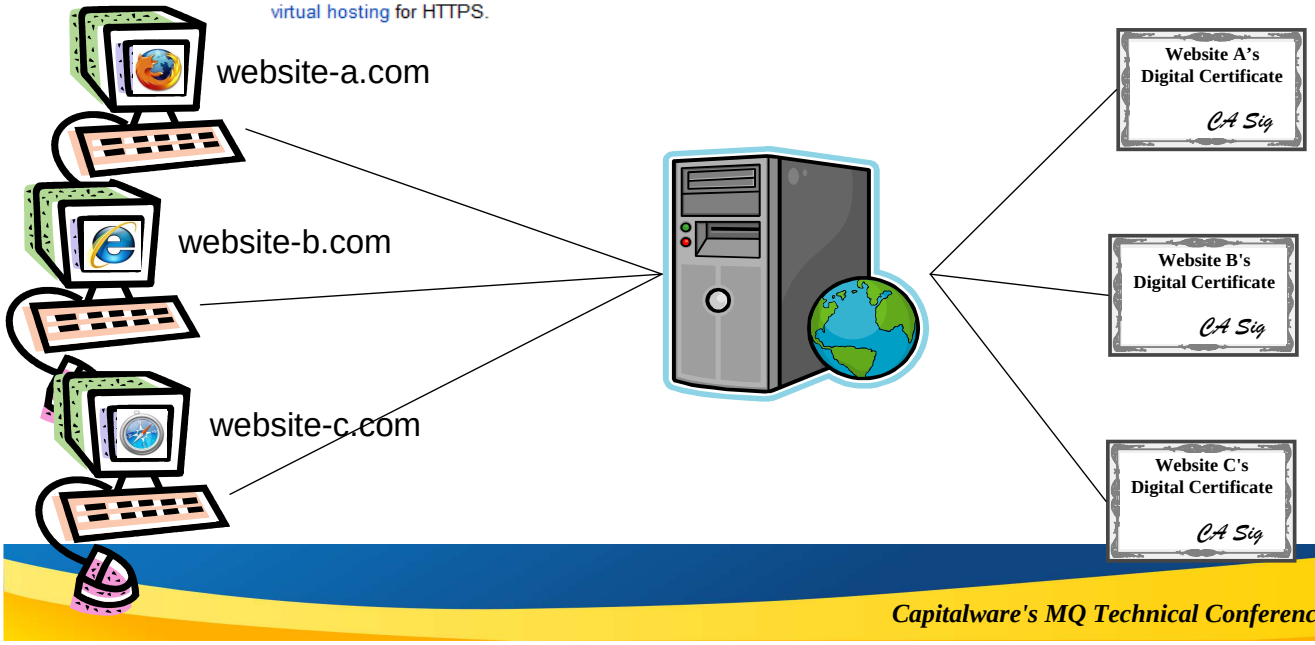
- The SSL/TLS handshake is done as the first thing on a channel, before any of the internal channel FAP flows. If you have ever pointed a web-browser with a https:// address at your MQ listener port, you'll know this. This means that the certificate is authenticated long before the channel name at the receiver end is known. This made it impossible to choose a certificate to be used for a receiver based on the channel name. The best that could have been done would have been to provide a different certificate per port number and have several different listeners running, each presenting a different certificate.
- Over time however, as SSL/TLS is used by more and more consolidated servers, think HTTP server farms and large application servers, it has become necessary to be able to separate the traffic that is going to a single server into differently authenticated groups.
- Enhancements to the TLS protocol allow the provision of information as part of the TLS handshake which can then be used to determine which certificate should be used for this particular connection.
- This enhancement is known as Server Name Indication (SNI).

Server Name Indication

Server Name Indication

From Wikipedia, the free encyclopedia

Server Name Indication (**SNI**) is an extension to the TLS protocol[1] that indicates what hostname the client is attempting to connect to at the start of the handshaking process. This allows a server to present multiple certificates on the same IP address and port number and hence allows multiple secure (HTTPS) websites (or any other Service over TLS) to be served off the same IP address without requiring all those sites to use the same certificate. It is the conceptual equivalent to HTTP/1.1 virtual hosting for HTTPS.

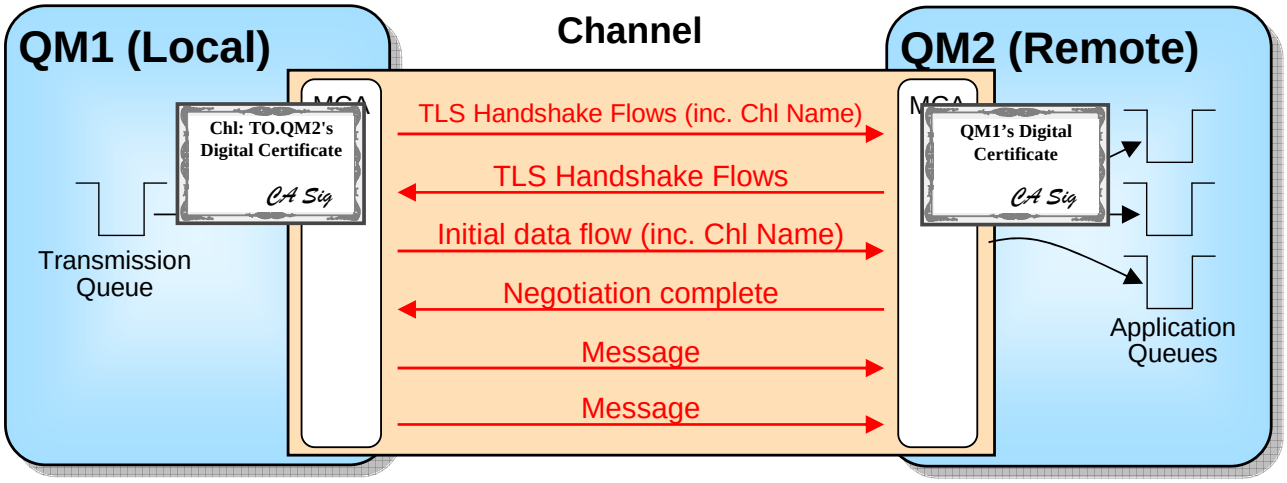


Server Name Indication – Notes

N O T E S

- Wikipedia provides a succinct summary of what Server Name Indication (SNI) is.
- The example on this page shows a use case where SNI would be used. We have three websites which each have their own certificate. When they were hosted on individual servers, then this was no problem, each web server has one certificate.
- Now let's think about what happens if we decide to consolidate those web sites onto a single server. How can we maintain the certificate correlation with the website. SNI allows this to be able to happen by providing a place in the TLS handshake for additional data to be flowed. This additional data is the hostname the browser was trying to connect to, thus allowing the certificate to be chosen based off that hostname.

Using Server Name Indication (SNI) with a channel name



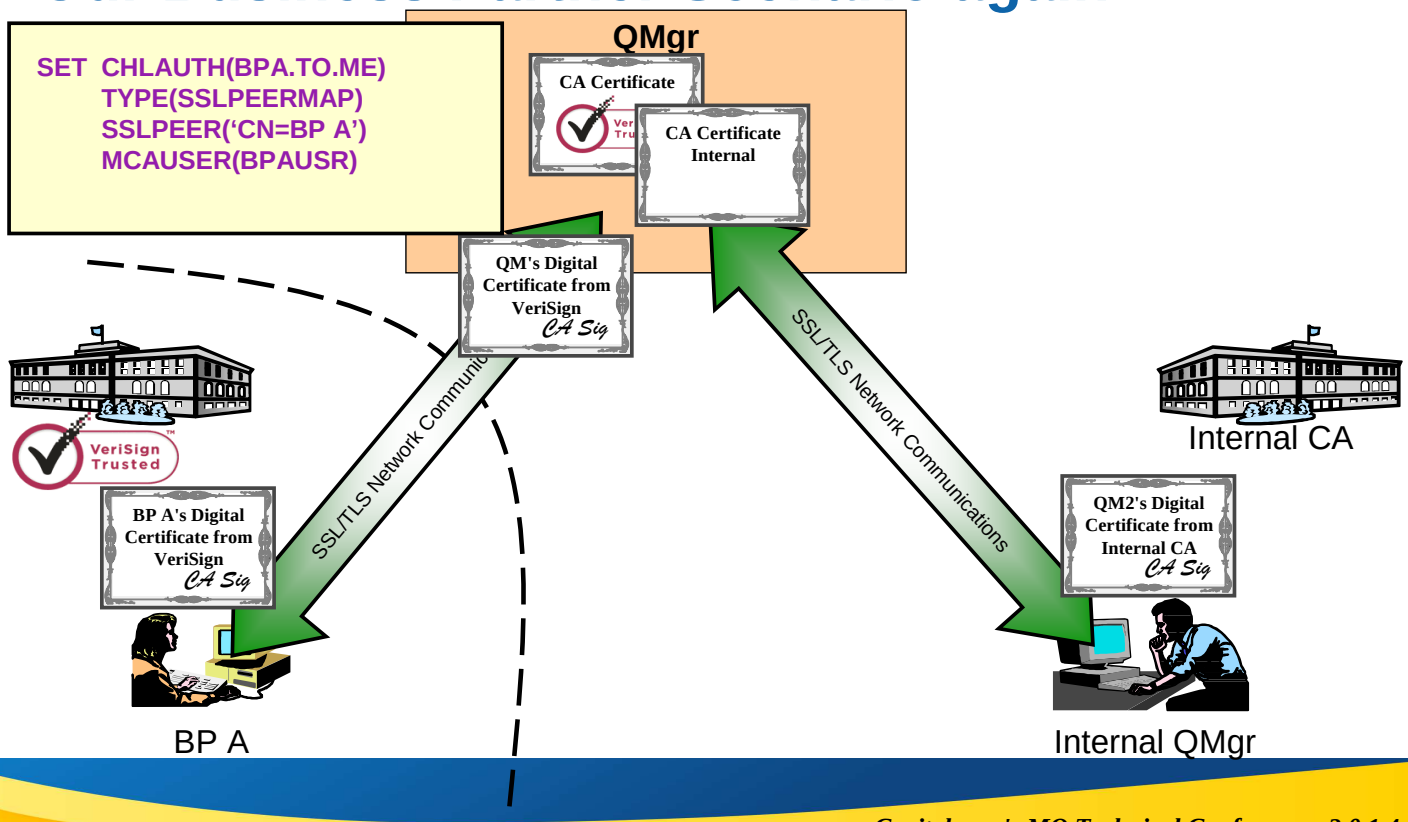
- **Both ends of the channel must be at the new release**
- **Only TLS can be used, no SSL**
  - Only certain cipherspecs will be able to supply this behaviour
- **JSSE doesn't yet support SNI**
  - So Java client can't make use of it
- **If old sender/client used, we'd only detect that we needed to supply a different certificate after completion of the handshake and will fail the connection, if it hasn't already failed due to using the wrong certificate!**

Using Server Name Indication (SNI) with a channel name

N O T E S

- WebSphere MQ V8 uses SNI to provide a channel name instead of a hostname. The sender (or client) end of the channel has been enhanced to put the channel name into the Server Name Indication (SNI) hint for the TLS Handshake.
- The receiver (or server-conn) end of the channel has been enhanced to retrieve the channel name from the SNI hint and select the appropriate certificate based on that information. It is worth nothing that the channel name is now flowing in the clear, although in a tamper-proof manner.
- There are some restrictions to using this feature as listed.
- A back-level queue manager upon receiving a TLS handshake containing SNI, will just ignore what is in the SNI (as it is defined as an optional extension) and use the normal certificate.
- If there are no channels defined on the queue manager with anything in the CERTLABL field, then SNI will not be used by the receiving end. This will leave the behaviour the same as prior releases for certificate selection.

Our Business Partner Scenario again

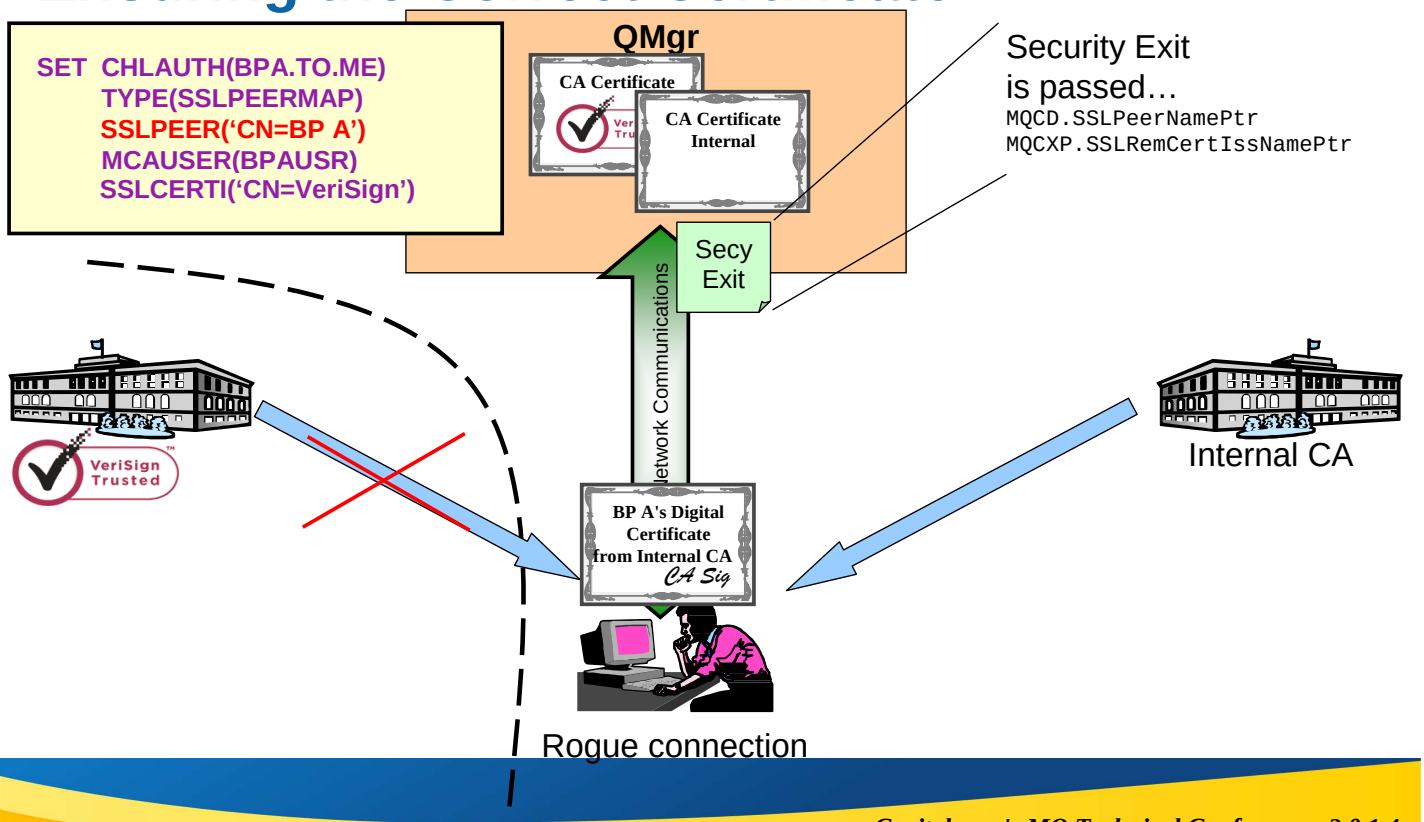


Our Business Partner Scenario again – Notes

N O T E S

- Let's look again at the business partner scenario again, but this time a little different, with one external CA and one internal CA.
- We've got the system set up so that we're using a Verisign certificate when talking to Business Partner A, and for the rest of our connections we have certificates created by our Internal CA. We've even got CHLAUTH rules in place to ensure that they are only allowed to connect to the queue manager over their appropriate channel.

Ensuring the Correct Certificate



Ensuring the Correct Certificate – Notes

N O T E S

- However, since we now accept certificates which come from two different Certificate Authorities (CAs) we can run foul of another issue.
- One of the benefits of CAs is that they guarantee not to issue the certificates with the same DN as another certificate that they have already issued. So a rogue connection could not obtain a certificate with the same DN as Business Partner A from VeriSign, because VeriSign has already issued one with that DN. Also, one would expect external CA's to do a few more checks than that and not issue certificates with other people's company names in them to people not from that company. However, an internal CA may not be so diligent. Some internal CAs may simply accept what the user requests as their DN, so our rogue could obtain a certificate with Business Partner A's DN from such a CA.
- The only way to solve this issue in the past was to use a security exit, since security exits are presented with both the issuer's and subject's Distinguished Name. However, we are trying to get away from people having to write exits for common security issues, and this very much falls into that category.
- In WebSphere MQ V8, we can solve this issue by using a new attribute on CHLAUTH rules which matches the issuer's DN – SSLCERTI. Our CHLAUTH rules can now be fully qualfied to use both SSLPEER (the subject's DN) and SSLCERTI (the issuer's DN).

Certificate DN Matching

- WebSphere MQ V5.3
  - `ALTER CHANNEL(APP1.SVRCONN) CHLTYPE(SVRCONN)`
    `SSLPEER('CN="Morag Hughson",O=IBM')`
    `MCAUSER('Morag')`
  - `ALTER CHANNEL(APP1.SVRCONN.EXTRA) CHLTYPE(SVRCONN)`
    `SSLPEER('CN="Graham Richards",O=IBM')`
    `MCAUSER('Graham')`

Need an extra channel object for different patterns

- WebSphere MQ V7.1
  - `SET CHLAUTH(APP1.SVRCONN) TYPE(SSLPEERMAP)`
    `SSLPEER('CN="Morag Hughson",O=IBM')`
    `MCAUSER('Morag')`
  - `SET CHLAUTH(APP1.SVRCONN) TYPE(SSLPEERMAP)`
    `SSLPEER('CN="Graham Richards",O=IBM')`
    `MCAUSER('Graham')`

Now can consolidate channel definitions

- IBM MQ V8
  - `SET CHLAUTH(APP1.SVRCONN) TYPE(SSLPEERMAP)`
    `SSLPEER('CN="Morag Hughson",O=IBM')`
    `SSLCERTI('CN="MQ Devt" CA,O=IBM')`
    `MCAUSER('Morag')`
  - `SET CHLAUTH(APP1.SVRCONN) TYPE(SSLPEERMAP)`
    `SSLPEER('CN="Graham Richards",O=IBM')`
    `SSLCERTI('CN="MQ Devt" CA,O=IBM')`
    `MCAUSER('Graham')`

Not available on channel object

Certificate DN Matching - Notes

N O T E S

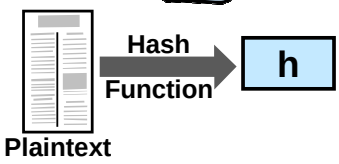
- In the original version of WebSphere MQ that supplied SSL support (WebSphere MQ V5.3) certificate DN matching could be done using the SSLPEER attribute on a channel definition. This was a single pattern per channel, and although it allowed wildcard matching the patterns you were matching on had to be fairly similar on any one channel.
- In WebSphere MQ V7.1 CHLAUTH rules were introduced that also allowed certificate DN matching and were more flexible because along with the wildcard matching that you had before, you also had the ability to add as many rules as you wanted to, to any one channel name.
- In IBM MQ V8, the CHLAUTH rules were extended further to allow you to match not only on the Subject's DN but also on the Issuer's DN allow even more control over recognising a partner's certificate. SSLCERTI is the CHLAUTH attribute for the Issuer's DN and it has not been added to the channel definition. It is recommended that you use CHLAUTH for all your certificate DN matching requirements.

Security ~~Problems?~~ Solutions with IBM MQ

- **Confidentiality**
  - Symmetric Key Cryptography

SSLCIPH(RC4_MD5_US)
SSLRKEYC(999 999 999)

- **Data Integrity**
  - Hash Function



- **Authentication**
  - Digital Certificates

SSLKEYR(QM1KEYRING)
CERTLABL('QM1Cert')
SSLPEER('O=IBM')
SSLCERTI('CN=MQ CA')
SSLCAUTH(REQUIRED)

  - Asymmetric Keys

  - Revocation status checking

SSLCRLNL(LDAPNL)

Security Solutions with IBM MQ - Notes

N O T E S

- Here we show three main security problems, eavesdropping, tampering and impersonation.
- We show the techniques that can be used to solve these problems. For eavesdropping, we have symmetric key cryptography; for tampering we have the hash function; and for impersonation we have digital certificates, asymmetric keys and certificate revocation.
- We have shown how WebSphere MQ makes use of these techniques to provide these solutions to these security problems. One can specify which symmetric key cryptography algorithm and which hash function to use by providing WebSphere MQ with a CipherSpec. Digital Certificates and Public Keys are found in a key repository with a label, both of which can be specified to WebSphere MQ. We can also check that we are talking to the partner we expect to be talking to by checking the Subject's and Issuer's Distinguished Name (DN) and can choose to authenticate both ends of the connection or only the SSL Server end of the connection. Also we can make use of certificate revocation lists or OCSP.

Summary of IBM MQ V8 SSL/TLS related features

- **Single Queue Manager Certificate**
  - `ALTER QMGR CERTLABL('My certificate name')`
- **Per Channel Certificate**
  - `ALTER CHANNEL … CERTLABL('This channel certificate')`
- **Certificate Matching**
  - ```
    SET CHLAUTH('*')
        TYPE(SSLPEERMAP)
        SSLPEER('CN=Morag Hughson')
        SSLCERTI('CN=IBM CA')
        MCAUSER('hughson')
    ```
    ```