Introduction to IBM MQ Advanced Message Security (AMS)

Lyn Elkins – ***elkinsc@us.ibm.com***
Mitch Johnson – ***mitchj@us.ibm.com***

Wildfire
Technical Hands-On Workshops
IBM zGrowth Team

Agenda

- What is MQ AMS?
- Key features
- Pre-requisites and runtime environment
- Logical architecture
- Components
- Installation & configuration
- Summary

Message Level Security - Requirements

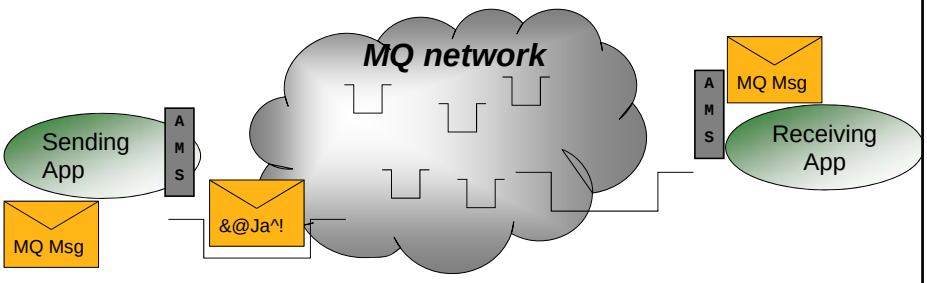
- Assurance that messages have not been altered in transit
  - When issuing payment information messages, ensure the payment amount does not change before reaching the receiver
- Assurance that messages originated from the expected source
  - When processing control messages, validate the sender
- Assurance that messages can only be viewed by intended recipient(s)
  - When sending confidential information

Why use message-level security?

- MQ networks : difficult to prove security of messages
  - Against message injection / message modification / unauthorized viewing
  - Prevalence of sub-contractors
  - Increasing levels of partnerships
- More and more data subject to standards compliance
  - Credit card data protected by PCI
  - Confidential government data
  - Personal information e.g. healthcare related
  - Data at rest, administrative privileges, etc
- Remember that base WebSphere MQ only provides message encryption when the MQ messages *are **in transit** over channels*. Without AMS, ***MQ messages have never been encrypted while they are sitting at rest in the queues with standard MQ****!*

What is IBM MQ Advanced Message Security?

- Provides security for MQ messages, end-to-end with no application changes
- It is a simple “add-on” product that enhances WebSphere MQ or IBM MQ
  - Base MQ security is not superseded
- Security policies are used to define the security level required which leverage X.509 certificates



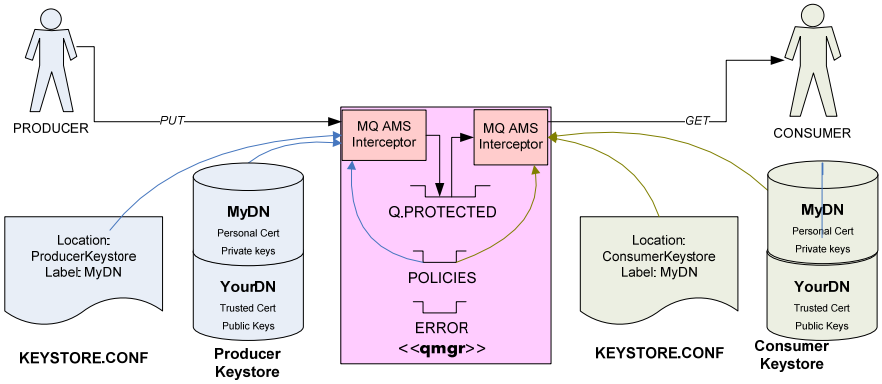
AMS Key Features

- Provides additional security to the security over what is provided by base MQ
- End-to-end security, message level protection
  - A security policy defines what protection should be applied to messages
  - AMS intercepts messages at “endpoints” and applies the policy
  - Well suited to point to point, can also protect publish/subscribe but...
  - … have to know the identity of the intended recipients ahead of operation
- Asymmetric cryptography used to protect each message
  - Integrity Policies prove message origin, content not changed
  - Privacy policies as per integrity plus each message encrypted with unique key
- Non-invasive
  - No code changes or re-linking of applications
- Administrative interfaces for policy management
  - Command line
  - MQ Explorer (Security Policies - now a default plugin)

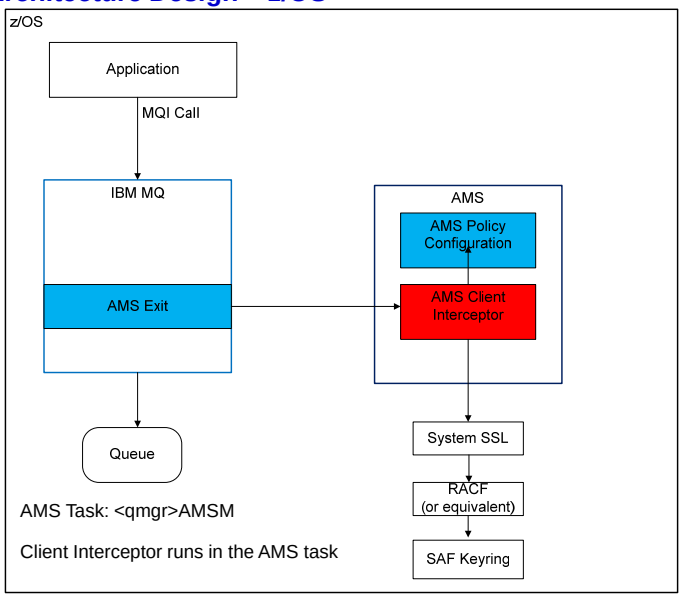
Environments supported

- MQ AMS functionality is implemented in “interceptors”
  - There are no long running processes or daemons (except in z/OS)
  - Existing MQ applications **do not require changes**
- Implemented as standard Queue Manager API exit on distributed, and “private” API exit on z/OS
- Three interceptors are provided:
  1. **MQ Server interceptor** for local (bindings mode) MQI API and Java applications.
  2. **MQ Client API interceptor** for remote (client mode) MQ API applications.
  3. **MQ Java client interceptor** for remote (client mode) MQ JMS and MQ classes for java applications (J2EE and J2SE).

Logical Architecture Design – Distributed Platforms



Logical Architecture Design – z/OS



WebSphere MQ Advanced Message Security – Security Features

- AMS is an optional component of MQ, not a replacement to base MQ security
- WebSphere MQ base review
  - Authentication (Local OS user id, SSL peer and CHLAUTH for channels)
  - Authorization (OAM and CHLAUTH on distributed, RACF on z/OS)
  - Integrity (SSL for channels)
  - Privacy (SSL for channels)
- WebSphere MQ Advanced Message Security
  - Integrity (End-to-end digital signing of messages)
  - Privacy (End-to-end message content encryption)

Message protection policies

- Created or updated or removed by
  - ***setmqspl*** command (invoke using CSQ0UTIL utility on z/OS)
  - MQ AMS plug-in for MQ Explorer (GUI) (distributed only)
- Policies are stored in queue `'SYSTEM.PROTECTION.POLICY.QUEUE'`
- Each protected queue can have only one policy
- For distributed queuing, protect the queue locally (source QM) as well as the remotely (target QM)
- Two types of policies:
  - Message Integrity policy
  - Message Privacy policy
- Display policies with command '`dspmqspl`'
- Messages not meeting policy requirements are placed in queue `'SYSTEM.PROTECTION.ERROR.QUEUE'`

Message integrity policy definition

- There are message signing algorithms: MD5, SHA1, SHA256, SHA384 and SHA512
- The list of authorized signers is optional
  - If no authorized signers are specified then any application can sign messages.
  - If authorized signers are specified then only messages signed by these applications can be retrieved and messages put in the queue by unauthorized signers will not be retrievable.
  - Messages from other signers are sent to the error queue.
- On z/OS, same setmqspl command and syntax used as SYSIN DD input program CSQ0UTIL

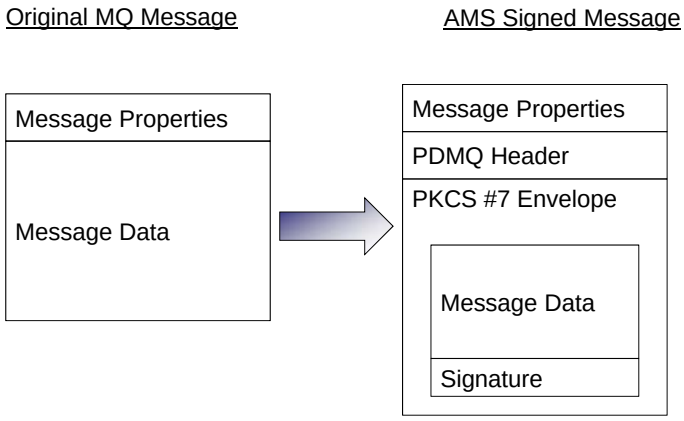
Syntax:

```
setmqspl
-m <queue_manager>
-p <protected_queue_name>
-s <MD5 | SHA1 | SHA256 |
   SHA384 | SHA512>
-a <Authorized signer DN>
-a <Authorized signer DN>
-r <Recipient signer DN>
```

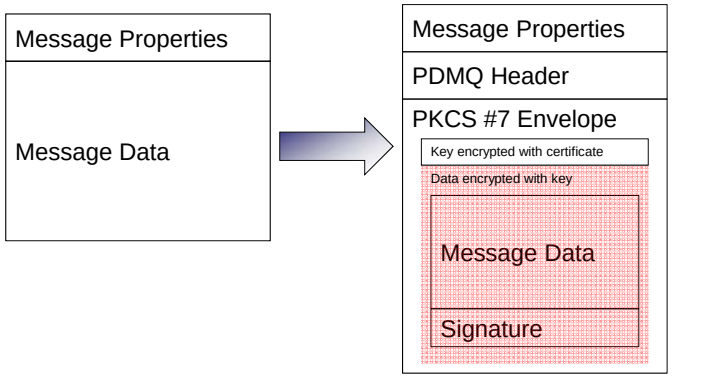
Example:

```
setmqspl -m QMZA
-p AMSDEMO.INTEGRITY.QUEUE
-s MD5
-a 'CN=user2,O=ibm,C=US'
-r 'CN=user2,O=ibm,C=US'
-r 'CN=user3,O=ibm,C=US'
```

MQ AMS : Integrity Message Format

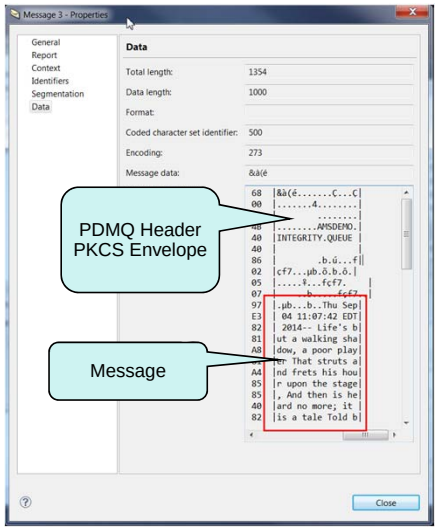


WebSphere MQ AMS : Privacy Message Format

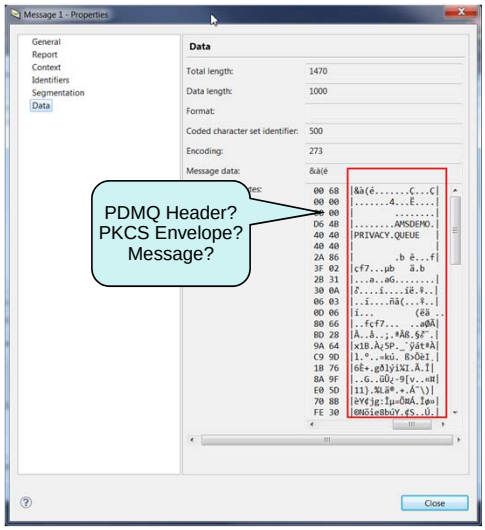


Encrypted message examples (access using queue aliases)

Message Integrity



Message Privacy



Message privacy policy definition

- Encryption algorithms: RC2, DES, 3DES, AES128 and AES256.
- Message privacy requires that encrypted messages are also signed.
- The list of authorized signers is optional.
- It is mandatory to specify at least one message recipient
- Messages retrieved by un-authorized recipients cause messages to be sent to the SYSTEM.PROTECTION. ERROR.QUEUE when dequeue attempted (destructive get not browse).

Syntax:

```
setmqspl
-m <queue_manager>
-p <protected_queue_name>
-s <MD5 | SHA1 | SHA256 | SHA384 | SHA512>
-e <encryption algorithm>
-a <Authorized signer DN1>
-a <Authorized signer DN2>
-r <Message recipient DN1>
-r <Message recipient DN2>
```

Example:

```
setmqspl -m QMZA
-p AMSDEMO.PRIVACY.QUEUE
-s MD5
-e RC2
-a 'CN=user1,O=ibm,C=US'
-r 'CN=user2,O=ats,C=US'
```

Keystores, KeyStores and X.509 certificates

- **Each MQ application** producing or consuming protected messages **requires access to a keystore/keyring** that contains a personal X.509 (v2/v3) certificate and the associated private key.
- The keystore/keyring and certificate is accessed by the MQ AMS interceptors.
- The keystore/keyring must contain trusted certificates to validate message signers or to obtain the public keys of encrypted message recipients
- Keystore/keyring can be the same as that used for MQ SSL
- Several types of keystore are supported (Distributed): CMS, JKS and JCEKS.
- On Distributed MQ, the IBM Key Management (iKeyman, part of GSKit) is provided to create and do simple management of local keystores
- On z/OS, standard SAF product (eg. RACF) used to create certificates which are SAF-managed and must be on a keyring named "`drq.ams.keyring`"
- 3rd party software is available from IBM (or others) to provide more robust, industrialisation of keystore maintenance.

MQ AMS configuration file - Distributed

- MQ AMS interceptors require a configuration file, eg. `KEYSTORE.CONF`, which contains:
  - Type of keystore: CMS, JKS, JCEKS
  - Location of the keystore.
  - Label of the personal certificate.
  - Passwords to access keystore and private keys (or `.sth` stash for CMS format)
- Interceptors locate the configuration file using one of the following methods:
  - Environment variable MQS_KEYSTORE_CONF=<path to conf file>.
  - Checking default locations and file names.
    - Platform dependent. For example in UNIX: "$HOME/.mqs/keystore.conf"

AMS z/OS installation

- Customization tasks for SYS1.PARMLIB
  - Update the Authorized Program Facility (APF) member (PROGxx) to add SDRQAUTH
  - Update the Program Properties Table (PPT) member (SCHEDxx) to add CSQ0DDRV
  - Review the common storage tracking member(DIAGxx) to ensure it allows for the allocation of common storage in user key
- Create a AMS task JCL procedure for the AMS started task
  - Based on sample member SCSQPROC(CSQ4AMSM)
- Define SAF profiles for started task, key rings and digital certificates
  - Set up the userid for the started AMS task and give SAF permissions,
  - Add SCSQAUTH and SCSQANLE to RACF program controlled list
- Note: userids that will be putting & getting protected messages will require:
  - An OMVS segment associated with their userid (or set default with FACILITY class, BPX.DEFAULT.USER)
  - SAF UPDATE permission for the FACILITY class, IRR.DIGTCERT.LISTRING

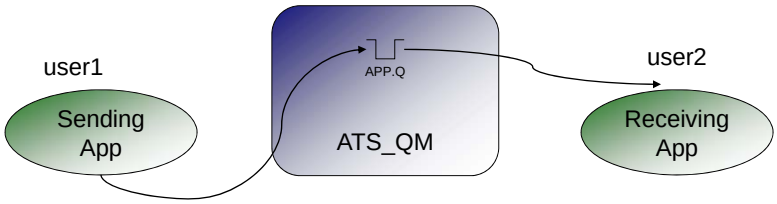
Sample RACF Commands required for AMS initial setup

Give AMS task's userid access to protected resources

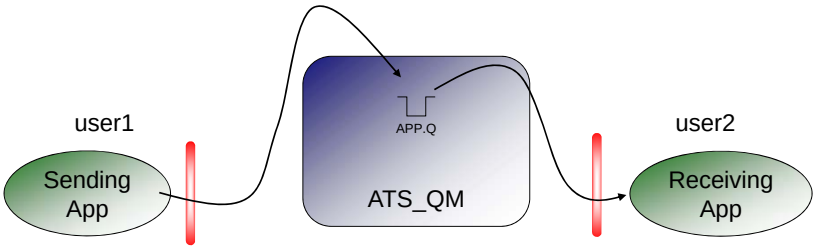
```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(QMZAUSR) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(USER1) ACCESS(UPDATE)
PERMIT BPX.SERVER CLASS(FACILITY) ID(QMZAUSR) ACCESS(READ)
PERMIT BPX.DAEMON CLASS(FACILITY) ID(QMZAUSR) ACCESS(READ)

SETROPTS RACLIST(FACILITY) REFRESH

RDEFINE SURROGAT BPX.SRV.* UACC(NONE)
PERMIT BPX.SRV.* CLASS(SURROGAT) ID(QMZAUSR) ACCESS(READ)

SETROPTS RACLIST(SURROGAT) REFRESH

RALTER PROGRAM * ADDMEM('MQ800.SCSQAUTH'//NOPADCHK)
RALTER PROGRAM * ADDMEM('MQ800.SCSQSANLE'//NOPADCHK)
SETROPTS WHEN(PROGRAM) REFRESH
```

"Authorize" AMS modules for OMVS processes

An existing MQ application – With No AMS protection

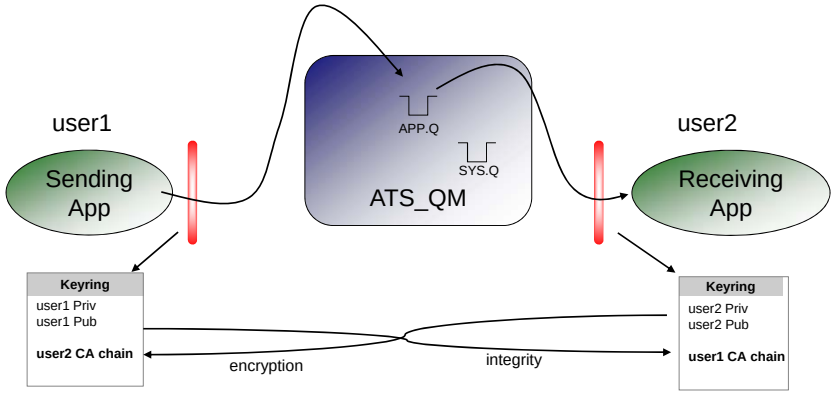


How to secure an existing MQ application - SPLCAP(ENABLED)



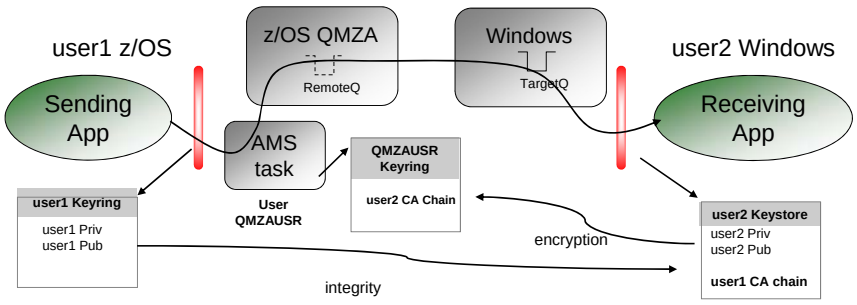
- Enable AMS on server

How to secure an existing MQ application – Create Digital Certificates



- Enable AMS component on server
- Create keyring with public / private key pairs
  a) connect sender's CA key chain to receiver's key ring for integrity
  b) connect receiver's CA key chain to sender's key ring for encryption

How to secure an existing MQ application – z/OS to Windows



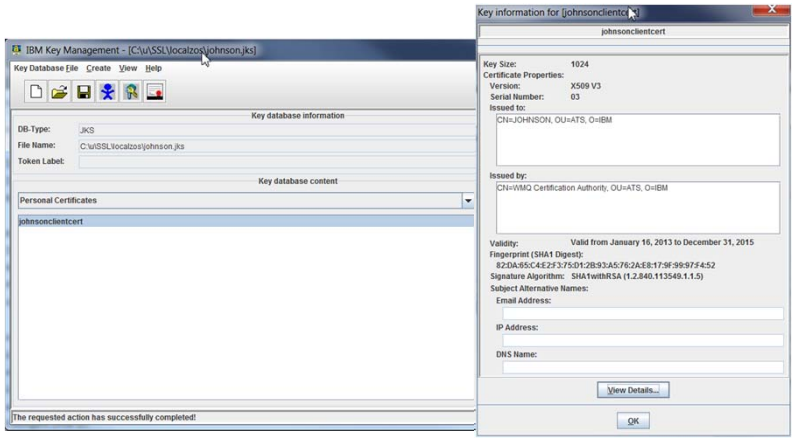
- Enable AMS component on servers
- Create keystores/keyrings with public / private key pairs
  a) Export and copy sender's CA key chain to receiver's keystore for integrity
  b) Export and copy receiver's CA key chain to key ring for encryption

AMS configuration on z/OS

- Define AMS system queues using the contents of CSQ4INSM in SCSQPROC
  - e.g.. by adding a new DD JCL statement to the CSQINP2 DD concatenation list
- Regenerate the queue manager's zPARM module after adding or changing the SPLCAP parameter in zPARM CSQ6SYSP macro to YES
- Add a new JCL procedure for the AMS main task, e.g. QMZAAMSM
  - Review and update as necessary the environment variables accessed by DD name ENVARS
  - Review and update as necessary the contents of the revoked certificate list accessed by DD name CRLFILE
- Create the required SAF (RACF) resources, keyrings, certificates, etc.
- Restart the queue manager and look for messages like:
  - `CSQY025I QMZA IBM WebSphere MQ AMS for z/OS is installed`
  - `CSQY027I QMZA CSQ0ERST AMS STARTING`
  - `CSQY028I QMZA CSQ0AMST AMS HAS STARTED`
- Ensure the AMS started task is active, e.g. QMZAAMSM

Certificate management - Distributed

- IBM MQ supplies iKeyman with GSKit
- Line-mode commands also available (eg. keytool,gsk7capicmd)
- On z/OS, RACF commands perform certificate management



RACF Command Example for z/OS

CA to validate certificates

```
/* Create a CA certificate good thru my retirement + 1 year */
RACDCERT CERTAUTH +
    GENCERT SUBJECTSDN(CN('AMSCA') O('ibm') C('us')) +
    WITHLABEL('AMSCA') KEYUSAGE(CERTSIGN) NOTAFTER(DATE(2031/12/31))

/* Connect the CA certificate to the AMSD task's key ring */
RACDCERT ID(QMZAUSR) ADDRING(drq.ams.keyring)
RACDCERT ID(QMZAUSR) CONNECT(CERTAUTH LABEL('AMSCA') RING(drq.ams.keyring)

RACDCERT EXPORT( LABEL('AMSCA') ) CERTAUTH FORMAT(CERTB64) DSN('QMZA.AMSCA.CERT')

/* Make a key ring for each user that will be putting or getting messages */
RACDCERT ID(USER1) ADDRING(drq.ams.keyring)
/* Create personal certificates userids */
RACDCERT ID(USER1) GENCERT SUBJECTSDN(CN('user1') O('ibm') C('us'))
    WITHLABEL('USER1') SIGNWITH(CERTAUTH LABEL('AMSCA')) -
    KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) NOTAFTER(DATE(2020/12/31))
RACDCERT ID(USER1) ALTER (LABEL('USER1')) TRUST

/* Connect each users personal certificates to their key ring */
RACDCERT ID(USER1) CONNECT(RING(drq.ams.keyring) -
    LABEL('USER1') DEFAULT USAGE(PERSONAL)

/* Connect the CA certificate to each user's key ring */
RACDCERT ID(USER1) CONNECT(RING(drq.ams.keyring) -
    LABEL('AMSCA') CERTAUTH USAGE(CERTAUTH)
/* Connect user's public key to AMS task's key ring */
RACDCERT ID(QMZAUSR) CONNECT(RING(drq.ams.keyring) -
    ID(USER1) LABEL('USER1') USAGE(SITE)
/*
```

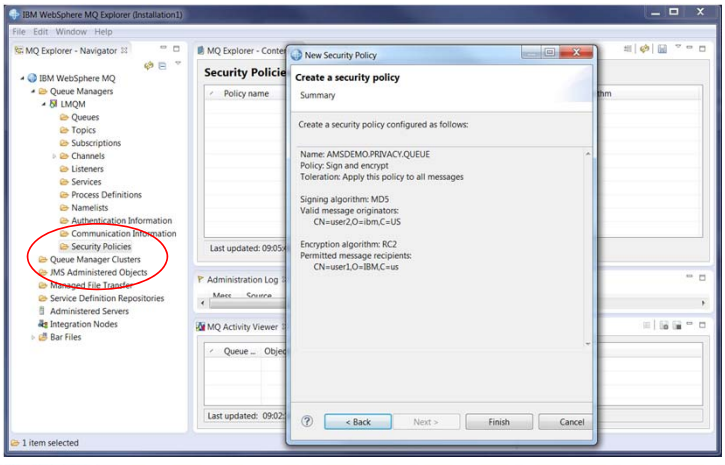
Export this CA so it can be imported into other keyrings or keystores

These commands would have to be repeated for each user

Queue policy definitions

- **Use either MQ Explorer (distributed queue managers only) or command line**

```
setmqspl -m LMQM -p AMSDEMO.PRIVACY.QUEUE -s MD5 -a
  "CN=user2,O=ibm,C=us" -e RC2 –r "CN=user1,O=ibm,C=us"
```



AMS CSQ0UTIL commands on z/OS

Execute AMS setmqspl command

```
//*******************************************************************
//*  Administer MQ Advanced Message Service (AMS) (CSQ0UTIL)        *
//*******************************************************************
//*
//CSQ40CFG EXEC PGM=CSQ0UTIL,
//         PARM='ENVAR("_CEE_ENVFILE_S=DD:ENVARS") /'
//STEPLIB  DD DSN=MQ800.SDRQLOAD,DISP=SHR
//         DD DSN=MQ800.SCSQANLE,DISP=SHR
//         DD DSN=MQ800.SCSQAUTH,DISP=SHR
//ENVARS   DD DSN=MQ800.SCSQPROC(CSQ40ENV),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 setmqspl -m QMZA -p AMSDEMO.PRIVACY.QUEUE
   -s MD5 -s "CN=user2,O=IBM,C=US"
   -e RC2 -r "CN=user1,O=ibm,C=US"
/*
```

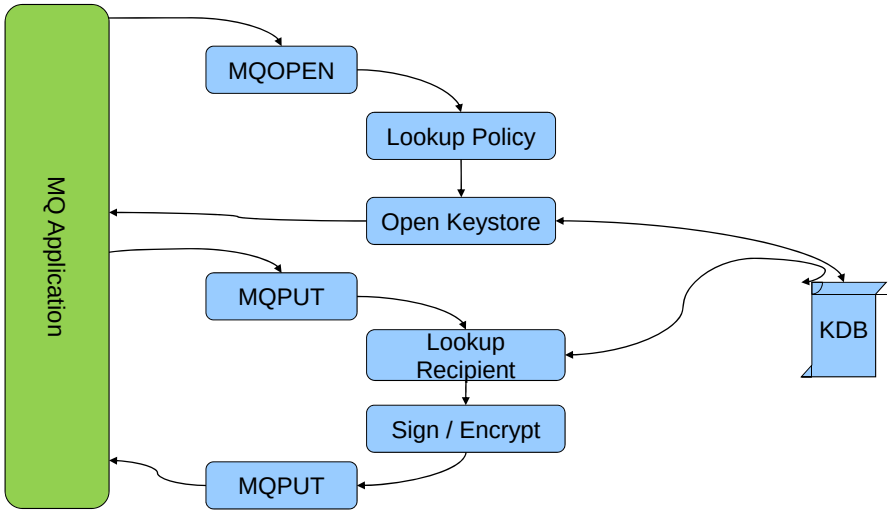
AMS Environment variables

```
_AMS_MSG_LEVEL=*.i
_AMS_MSG_FOLDING=NO
_AMS_INIT_THREADS=20
_AMS_MAX_THREADS=100
TZ=EST5EDT
```

Other useful AMS commands

```
dspmqspl -m QMZA -p AMSDEMO.PRIVACY.QUEUE         /* Displays policy
setmqspl -m QMZA -p AMSDEMO.PRIVACY.QUEUE -remove /* Removes policy
```


MQ AMS Process



Error handling

- AMS returns a RC=2063 if the application tries to access (MQGET) a message for which it is not authorized
  - CSQ0209E (user1) Message for AMSDEMO.INTEGRITY.QUEUE sent to error queue SYSTEM.PROTECTION.ERROR.QUEUE, reason 2063
- For destructive MQGET requests, the message is also transferred to the `SYSTEM.PROTECTION.ERROR.QUEUE.` The original message remains there with a DLQ header for administrative handling.

SSL errors on z/OS generate additional information

- CSQ0216E (USER1) Data unprotection failed processing object "pkcs7 enveloped data message", return code 12, reason 03353033.
- Additional information: function gsk_read_enveloped_data_content failed with x3353033, 00000008.
- CSQ0209I (USER1) Message for AMSDEMO.PRIVACY.QUEUE sent to error queue SYSTEM.PROTECTION.ERROR.QUEUE, reason 2063.
- The following is from *Cryptographic Services System Secure Sockets Layer Programming, SC24-5901* and provides an explanation of the gsk (Global Security Toolkit) return code.

03353033 Recipient certificate not found.

Explanation: A recipient certificate is not found while creating or processing an enveloped message.

User response: Provide at least one recipient certificate.

Known limitations today

- The following WebSphere MQ options are not supported
  - Publish/subscribe.
  - Channel data conversion.
  - Distribution lists.
  - Application message segmentation
  - The use of non-threaded applications using API exit on HP-UX platforms.
  - WebSphere MQ classes for .NET in a managed mode (client or bindings connections).
  - Message Service client for .NET (XMS) applications.
  - Message Service client for C/C++ (XMS supportPac IA94) applications.
  - WebSphere MQ queues processed by the IMS Bridge.Note: WebSphere MQ AMS is supported on CICS Bridge queues. You should use the same user ID to MQPUT (encrypt) and MQGET (decrypt) on CICS Bridge queues.
- All Java™ applications are dependant on IBM® Java Runtime. WebSphere MQ AMS does not support any JRE provided by other vendors.
- Users should avoid putting two or more certificates with the same Distinguished Names in a single keystore file because the WebSphere MQ Advanced Message Security interceptor's functioning with such certificates is undefined.
- Note that
  - message length will increase
  - AMS usage will increase CPU requirements

Summary

MQ Advanced Message Security

- Protects message integrity and/or privacy
- Supports WMQ V7 and IBM MQ V8
- Supports MQ Server, MQ Client and JMS
- “Light weight” product - No pre-requisites, easy installation, easy configuration
- Existing MQ applications do not require changes

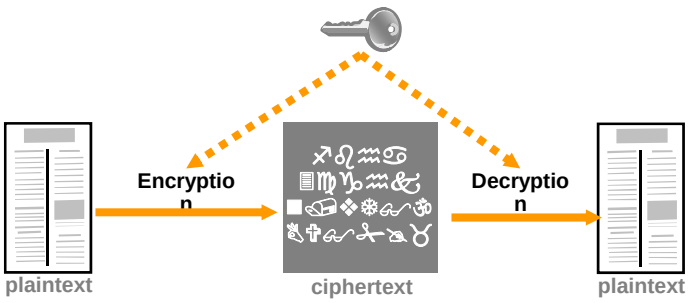
Additional Information

- MQ AMS Knowledge Center at: http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.pro.doc/q001010_.htm
- MP1J: WebSphere MQ V8.0 for z/OS Performance Report
  - http://www-01.ibm.com/support/docview.wss?uid=swg24038347
    - Protection isn't cheap; you either need it or you don't!
    - Cost is in CPU as well as management
    - Message size increase

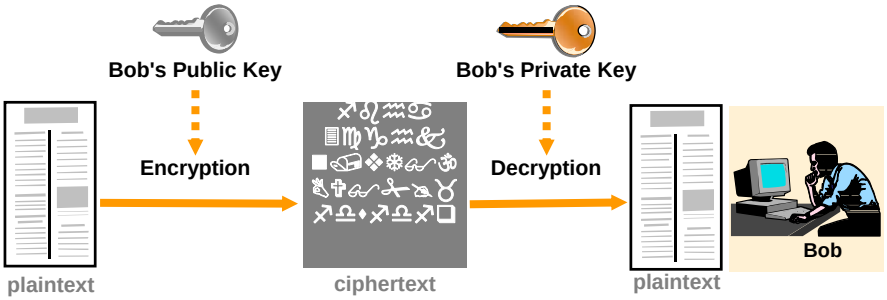
Cryptography Choices

- Symmetric Key
  - Single secret key
  - Relatively fast
  - Poses key distribution challenges when faced with large numbers of senders/receivers
  - The key has to be known by the sender and receiver
- Asymmetric Keys
  - Private & Public key pairing
  - Message encrypted with one key can only be decrypted by the other one
  - Slower than symmetric key cryptography
  - Asymmetric Keys can be used to solve the key distribution challenges associated with symmetric keys

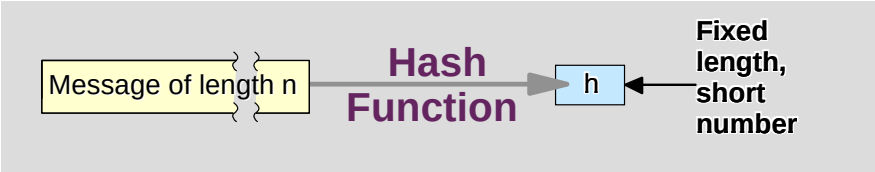
Symmetric Key Cryptography



Asymmetric Key Cryptography

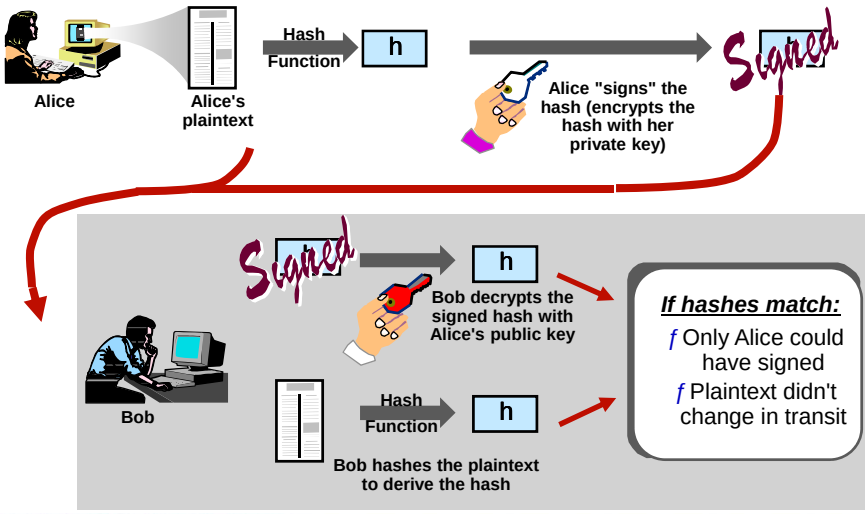


Hash Functions



- Hash Function
  - Computes the message MAC (Message Authentication Code)
  - Easy to compute
  - Very difficult to reverse
  - Computationally infeasible to find two messages that hash to the same value

Digital Signatures



AMS encryption performance on z/OS

- Protection isn't cheap; you either need it or you don't!
- Cost is in CPU as well as management
- Cryptographic CoProcessors are used for z/OS
- Relative costs for AMS V7.0.1:

	Cost MQOPEN+MQCLOSE	Cost MQPUT+MQGET
No AMS	111	60
AMS configured, unprotected queue	320	160
AMS configured, MD5 signature & RC2 encryption, 1 recipient	5000	2200

For additional details, see "Performance of AMS V7.0.1 on z/OS", http://www.ibm.com/support/docview.wss?uid=swg27019944