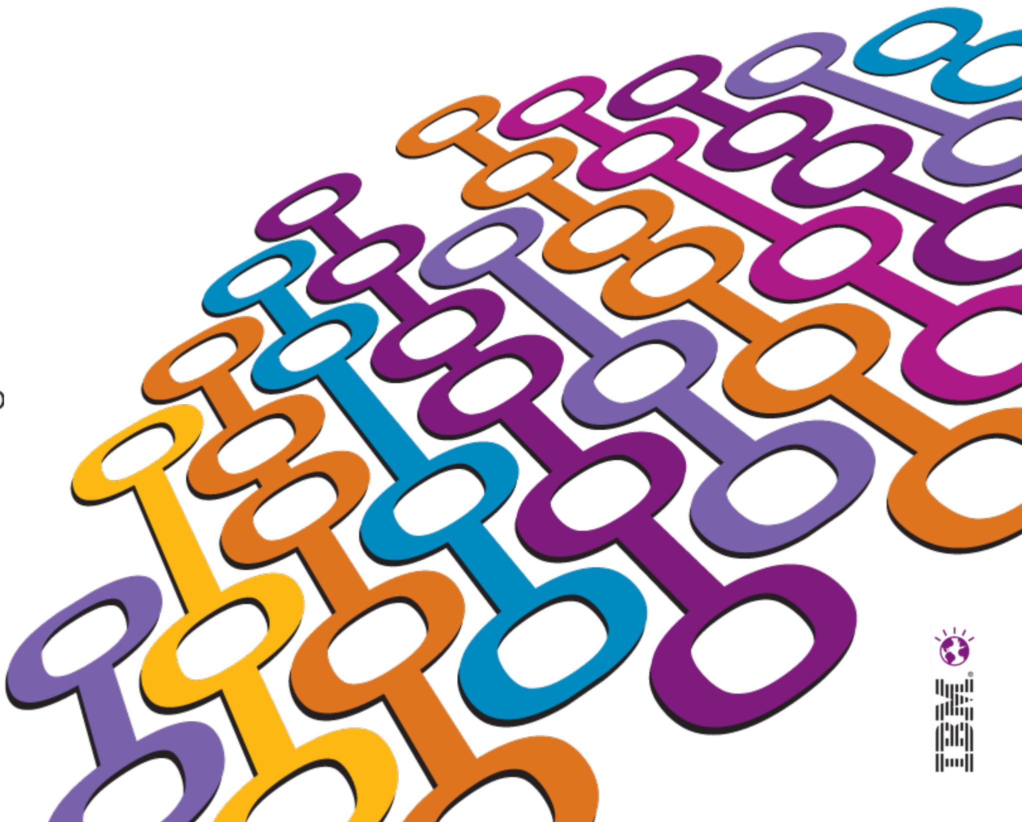
Securing Your Messages

T.Rob Wyatt
WebSphere Connectivity & Integration Product Manager, Security
t.rob.wyatt@us.ibm.com

Impact2012
The Premier Conference for Business and IT Leadership
Innovate. Transform. Grow.

Session 1578



Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Agenda

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

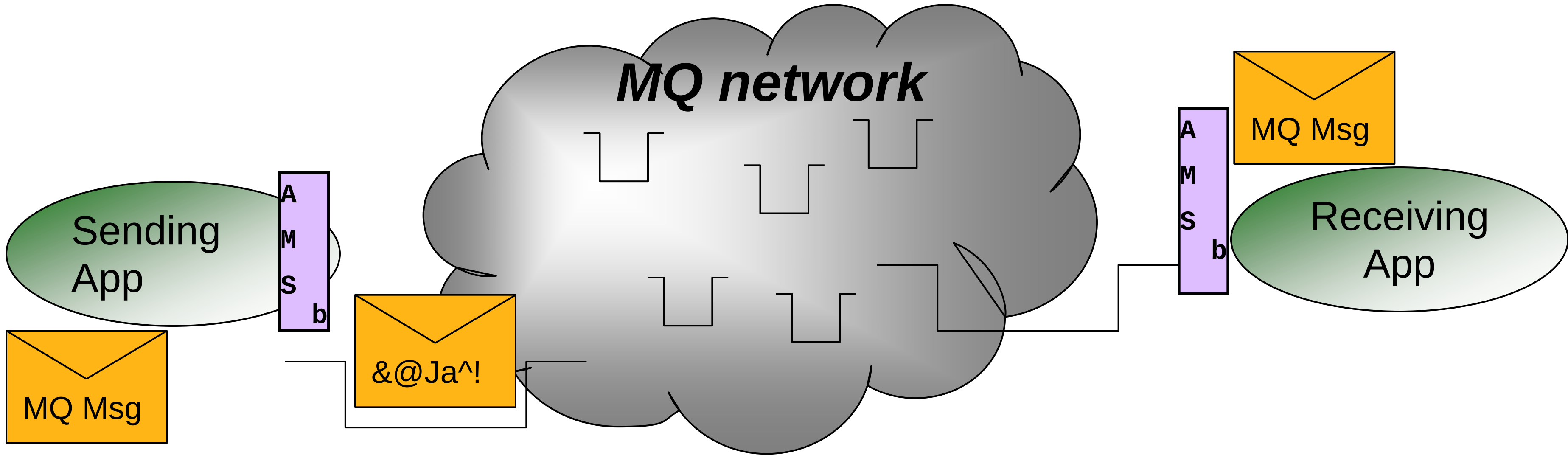
Why use message-level security?

- Base WebSphere MQ networks
  - Authentication and authorization is scoped to the connection
  - SSL/TLS channels provide additional connection-scoped security
  - Channel context setting provides some per-message authorization
    - But based on unauthenticated MQMD.UserID
- WMQ AMS complements WMQ's connection-level security
  - Provides authentication, authorization and accoutability scoped at the message level
- Increasing impact of regulatory compliance
  - Payment Card Industry Data Security Standard (PCI-DSS)
  - Health Insurance Portability & Accountability Act (HIPAA)
  - European Union Privacy Directive
  - FIPS, Suite-B, FISMA
- Provide additional security for Command & Control traffic
- Any time many identities are aggregated over a single connection

What is MQ AMS?

WebSphere MQ Advanced Message Security V7.0.1

- New product, announced and available Oct 5, 2010
- Fix Pack 1 released on March 2011
- Provides security for MQ messages, end-to-end with no application changes
- Simple “add-on” component that enhances WebSphere MQ v6 or v7.x
- Security requirements are expressed as user-definable policies
- AMS leverages X.509 certificates



AMS Key Features

- Secures sensitive or high-value MQ messages
  - Privacy via message content encryption
  - It leverages digital certificates (X.509) and Public Key encryption to protect MQ messages
- Detects and removes rogue or unauthorized messages before they are processed by receiving applications
  - Authentication via certificate *above and beyond* operating system
  - Authorization to queue *above and beyond* MQ OAM or SAF
- Verifies that messages are not modified between sender and receiver
  - Message Integrity via digital signature of message content
- Protects messages not only when they flow across the network but when they are at rest in queues
- Messages from existing MQ applications are transparently secured using "interceptors"
  - No application changes are necessary
- No pre-requisite products other than MQ
- Successor to WebSphere MQ Extended Security Edition (ESE)

Platforms supported
(5724-Z94 for Distributed, 5655-W50 for z/OS)

- Windows (32 & 64-bit, XP Pro, Server 2003, Server 2008, Vista)
- AIX for System p (v5.3, v6.1)
- HP-UX Itanium & PA-RISC (11i v2 & v3)
- Linux for System p (64-bit, RHEL v5 & v5, SLES v9, v10, v11)
- Linux for System x (32 & 64-bit, RHEL v5 & v5, SLES v9, v10, v11)
- Linux for System z (64-bit, RHEL v5 & v5, SLES v9, v10, v11)
- Solaris for Intel X86 (64-bit, v10)
- Solaris for Sun SPARC (64-bit, v9 & v10)
- z/OS for System z (z/OS v1.8) (IBM SSL v1.8 is also required)

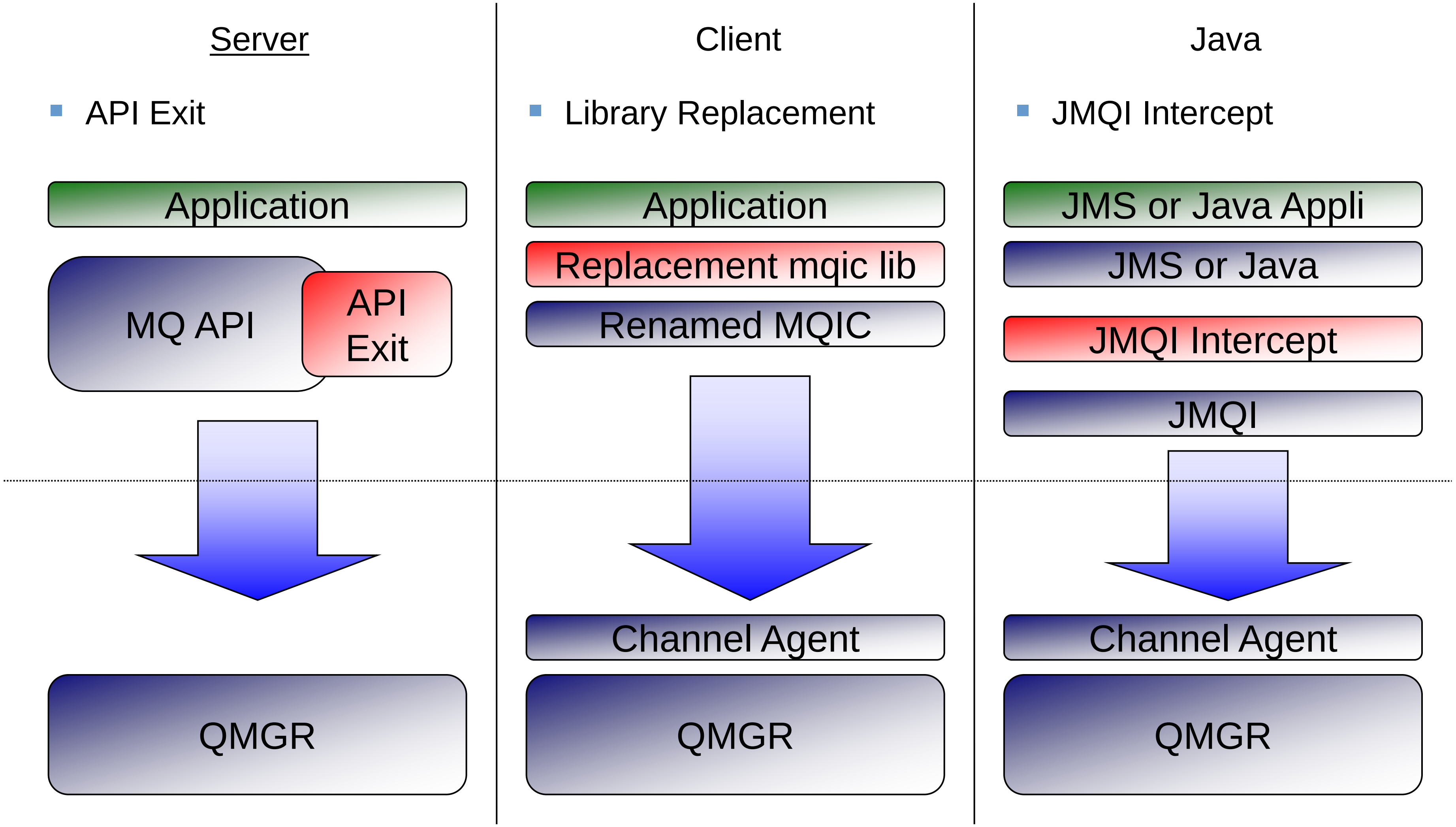
For complete details, see:
http://www.ibm.com/software/integration/wmq/advanced-message-security/reqs/

Environments supported

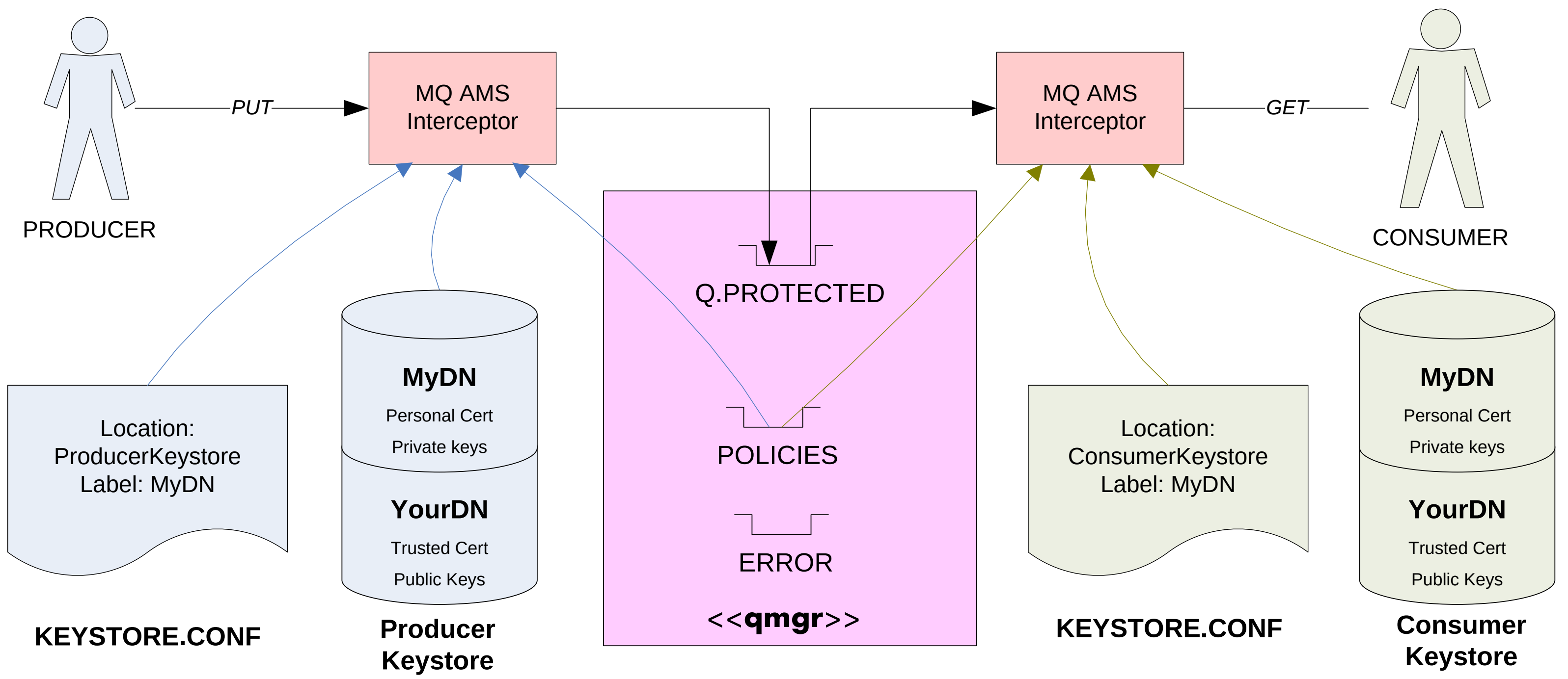
- MQ AMS functionality is implemented in “interceptors”
  - There are no long running processes or daemons (except in z/OS)
  - Existing MQ applications **do not require changes**
- Three interceptors are provided:

1. **MQ Server interceptor** for local (bindings mode) MQI API and Java applications.
   - Implemented as standard QM API exit on distributed, and “private” API exit on z/OS
   - Requires MQ v6.0.2.8 or 7.0.0.1 as well as GSKit 7.0.4.23 (minimum versions)
   - Note that MQ v7 is required for the AMS MQ Explorer plugin
2. **MQ Client API interceptor** for remote (client mode) MQ API applications.
   - MQ AMS interceptor imbedded in MQ client code
   - Requires MQ v6.0.2.8 or 7.0.1.1 as well as GSKit 7.0.4.23 (minimum versions)
3. **MQ Java client interceptor** for remote (client mode) MQ JMS and MQ classes for java applications (J2EE and J2SE).
   - MQ AMS interceptor imbedded in MQ java client code.
   - Requires MQ Java v7.0.1 as well as IBM Java 1.4..2 (minimum versions)

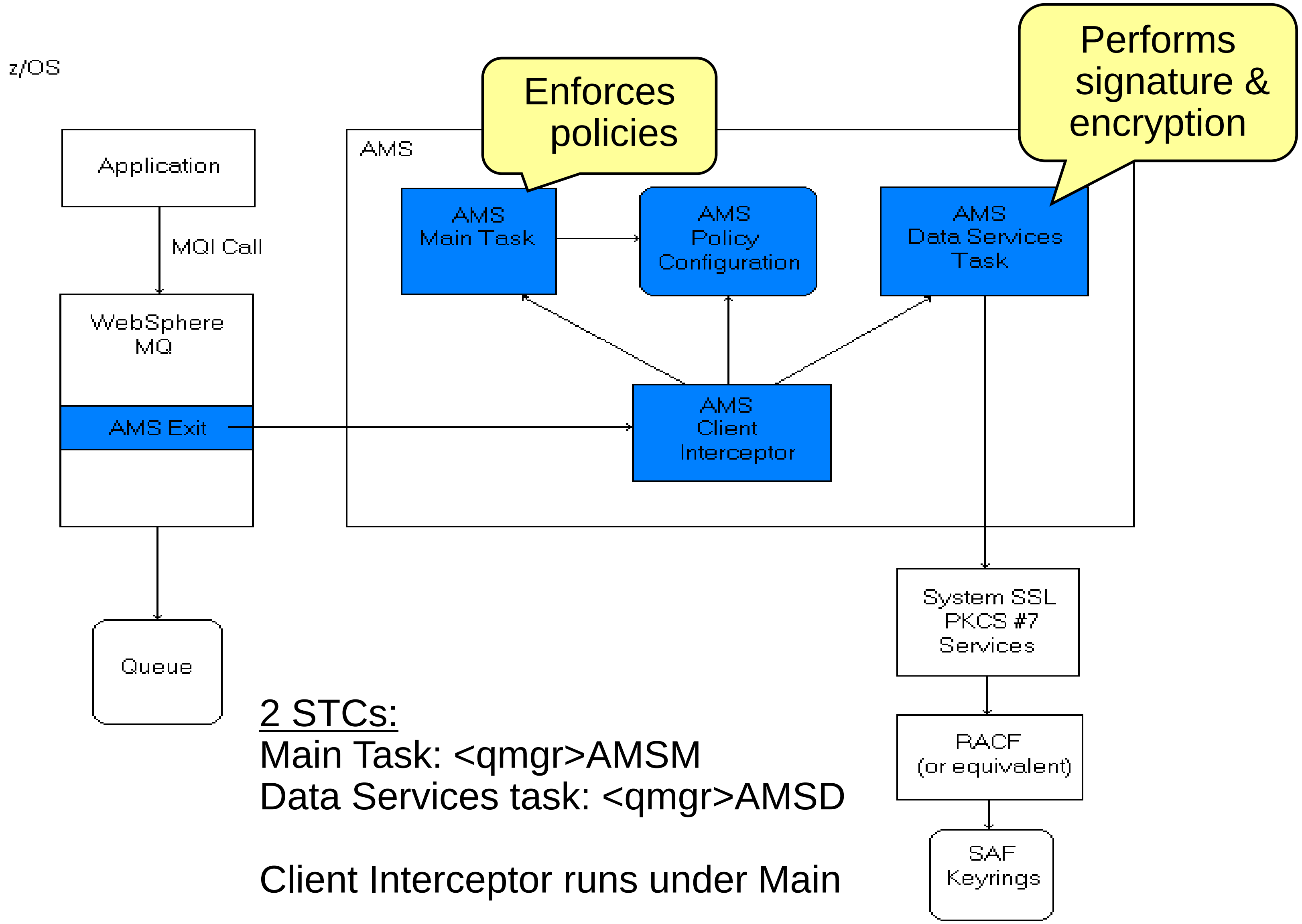
Interceptors



Logical Architecture Design
Distributed Platforms



Logical Architecture Design – z/OS



<u>2 STCs:</u>
Main Task: <qmgr>AMSM
Data Services task: <qmgr>AMSD

Client Interceptor runs under Main

Message protection policies

- Created or updated or removed by command ‘`setmqspl`’
  - or by MQ AMS plug-in for MQ Explorer (GUI)
- 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 remote (target QM)
- Two types of policies:
  - Message Integrity policy
  - Message Privacy policy
- Display policies with command ‘`dspmqspl`’
- “Compromised messages” in queue `‘SYSTEM.PROTECTION.ERROR.QUEUE’`
- Extra queue on z/OS `‘SYSTEM.PROTECTION.SYNC.QUEUE’`

Message integrity policy definition

- There are two message signing algorithms: SHA1 and MD5
- 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.
  - Messages from other signers are sent to the error queue
- On z/OS, same setmqspl program and parms used as SYSIN DD for PGM=DRQUTIL

Syntax:

```
setmqspl
-m <queue_manager>
-p <protected_queue_name>
-s <SHA1 | MD5>
-a <Authorized signer DN1>
-a <Authorized signer DN2>
```

Example:

```
setmqspl -m MYQM
-p MY.Q.INTEGRITY
-s SHA1
-e NONE
-a 'CN=cfarkas,O=ibm,C=FR'
```

Message privacy policy definition

- Signature algorithms: MD5, SHA1, SHA256*, SHA384* or SHA512*
- Encryption algorithms: RC2, DES, 3DES, AES128 and AES256
  - Encrypted messages are always signed
- The list of authorized signers is optional
- It is mandatory to specify at least one message recipient
- Retrieved messages which do not meet AMS policy sent to the SYSTEM.PROTECTION. ERROR.QUEUE
  - Eg: Policy contains authorized signer list and sender is not on it

Syntax:

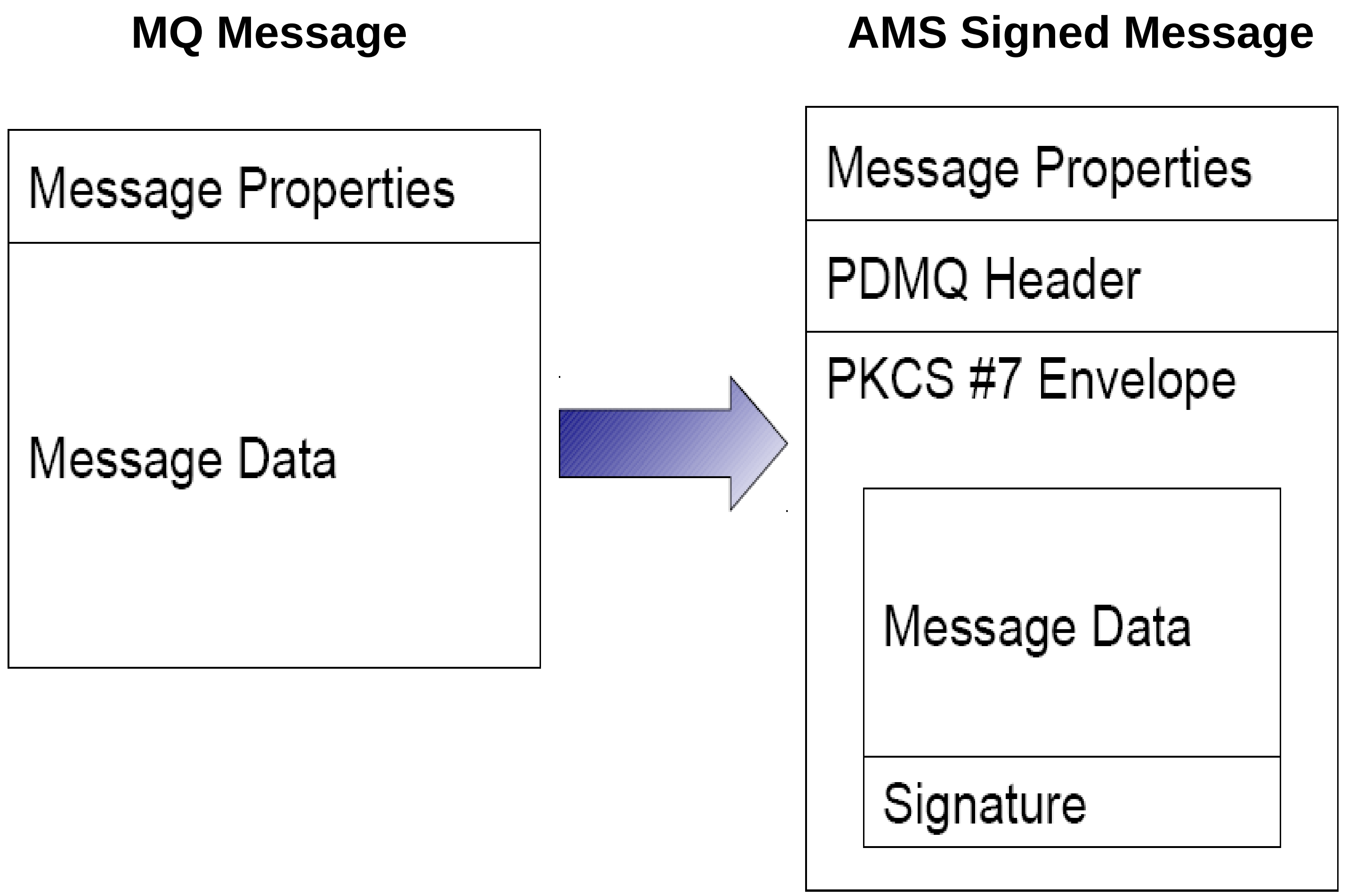
```
setmqspl
-m <queue_manager>
-p <protected_queue_name>
-s <SHA1 | MD5>
-e <encryption algorithm>
-a <Authorized signer DN1>
-a <Authorized signer DN2>
-r < Message recipient DN1>
-r < Message recipient DN2>
```

Example:

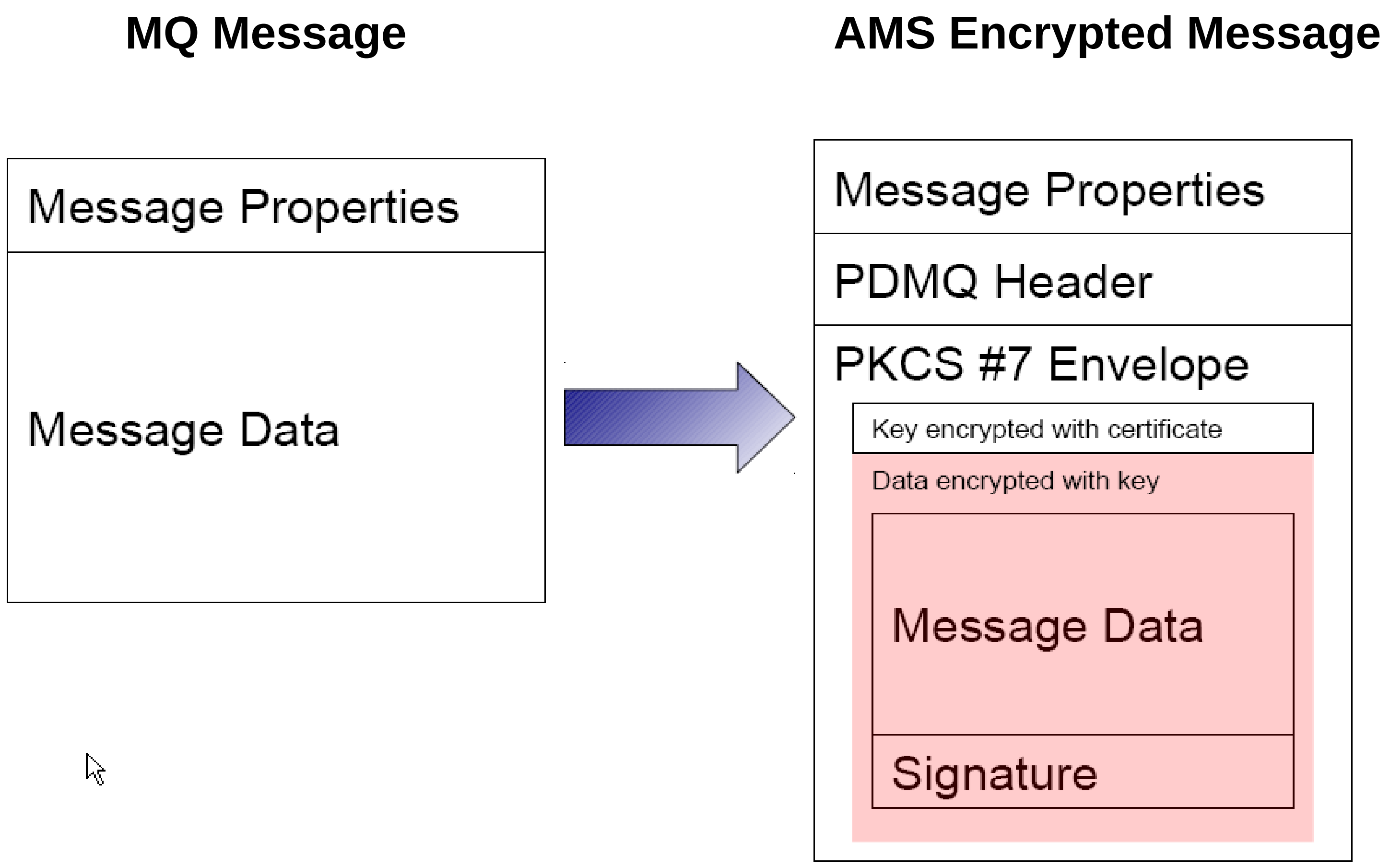
```
setmqspl -m MYQM
-p MY.Q.PRIVACY
-s SHA1
-e AES128
-a 'CN=carl,O=ibm,C=US'
-r 'CN=ginger,O=catunion,C=JP'
-r 'CN=saadb,OU=WBI,O=IBM,C=FR'
```

* Note: SHA-2 algorithms available in v7.0.1.2 and higher

Integrity message format



Privacy message format



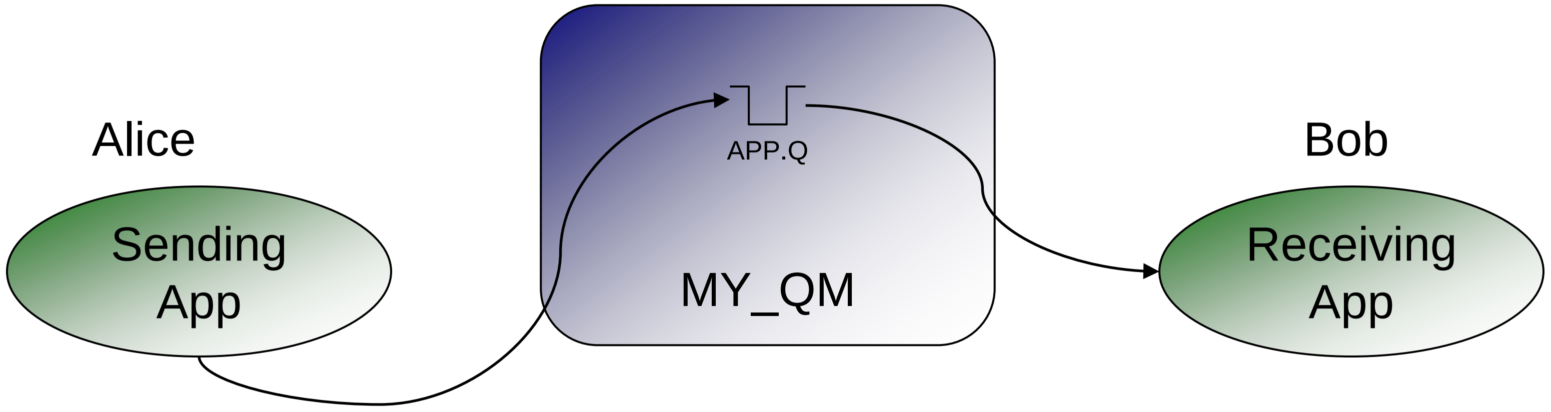
Keystores and X.509 certificates

- **Each MQ application** producing or consuming protected messages **requires access to a keystore** that contains a personal X.509 (v2/v3) certificate and the associated private key.
- The keystore and certificate is accessed by the MQ AMS interceptors.
- The keystore must contain trusted certificates to validate message signers or to obtain the public keys of encrypted message recipients
- Keystore 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. For the IBM Tivoli Key Lifecycle Manager Tivoli, see: http://www.ibm.com/software/tivoli/products/key-lifecycle-mgr/

MQ AMS configuration file

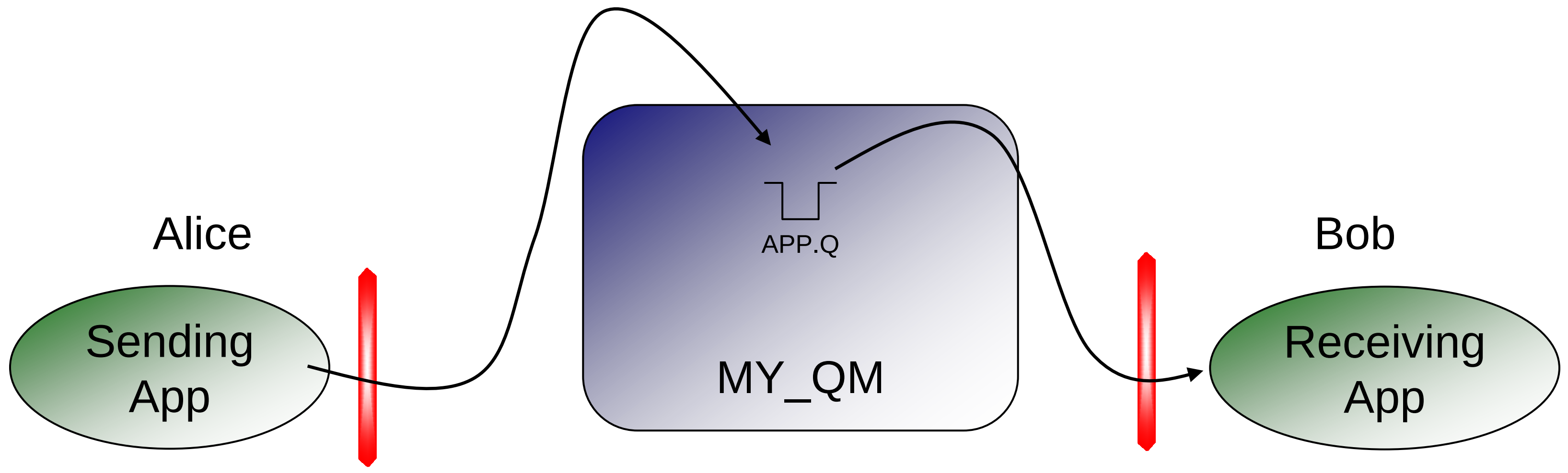
- MQ AMS interceptors require a configuration file, eg. `KEYSTORE.CONF`, which contains:
  - Type of keystore: CMS, JKS, JCEKS
  - Location of the keystore.
  - Label of the personal certificate.
  - Passwords to access keystore and private keys (or `.sth` stash for CMS format)
- Interceptors locate 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”

WebSphere MQ AMS – install/config example, 0



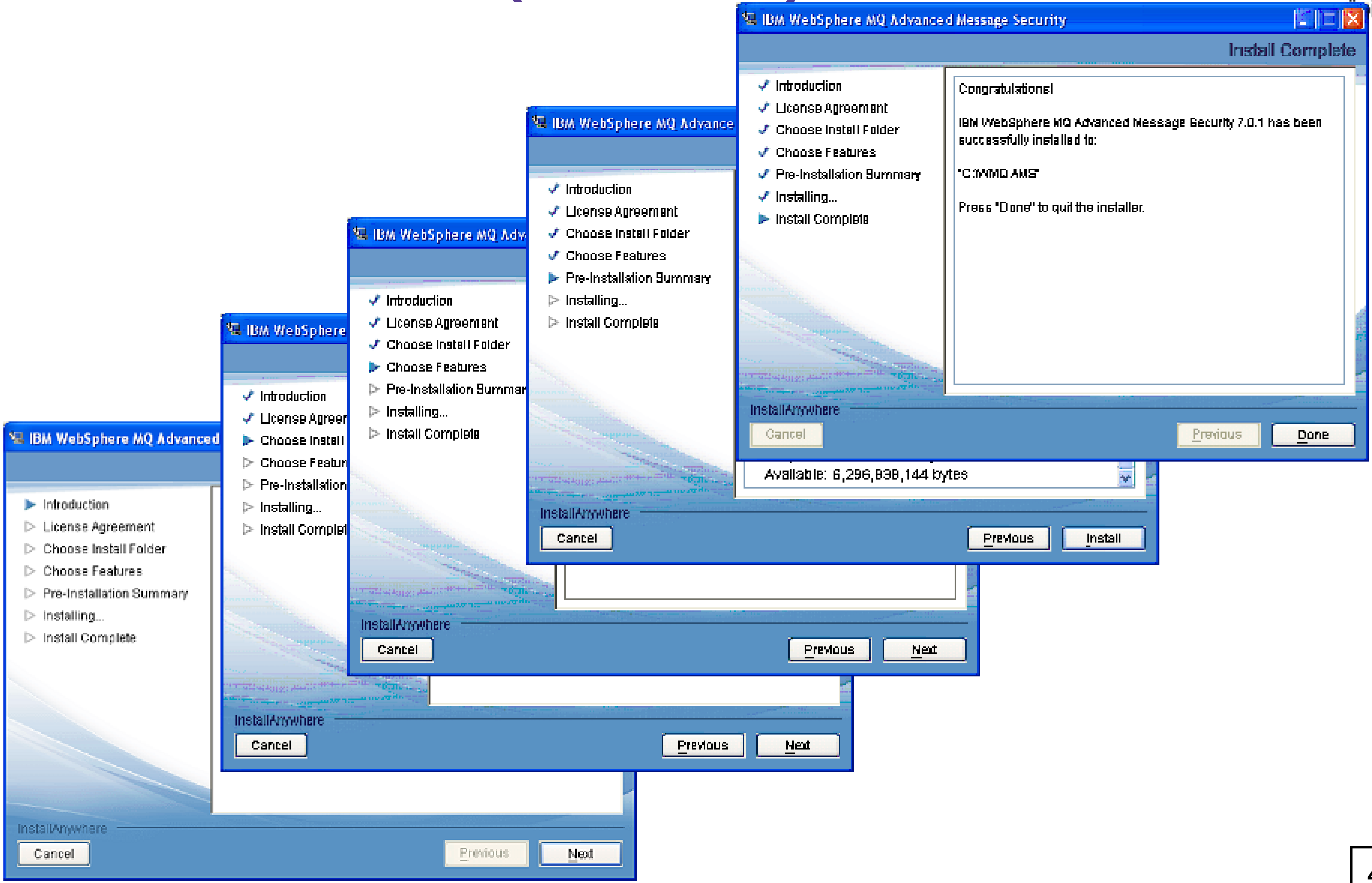
For a good step-by-step guide, see the AMS InfoCenter at
`http://publib.boulder.ibm.com/infocenter/mqams/v7r0m1/index.jsp`
and search for “Quick start”

WebSphere MQ AMS – install/config example, 1



1.Install and configure AMS Interceptor

AMS installation (Windows)



IBM WebSphere MQ Advanced Message Security

Install Complete

Congratulations!

IBM WebSphere MQ Advanced Message Security 7.0.1 has been successfully installed to:

"C:\WMQ AMS"

Press "Done" to quit the installer.

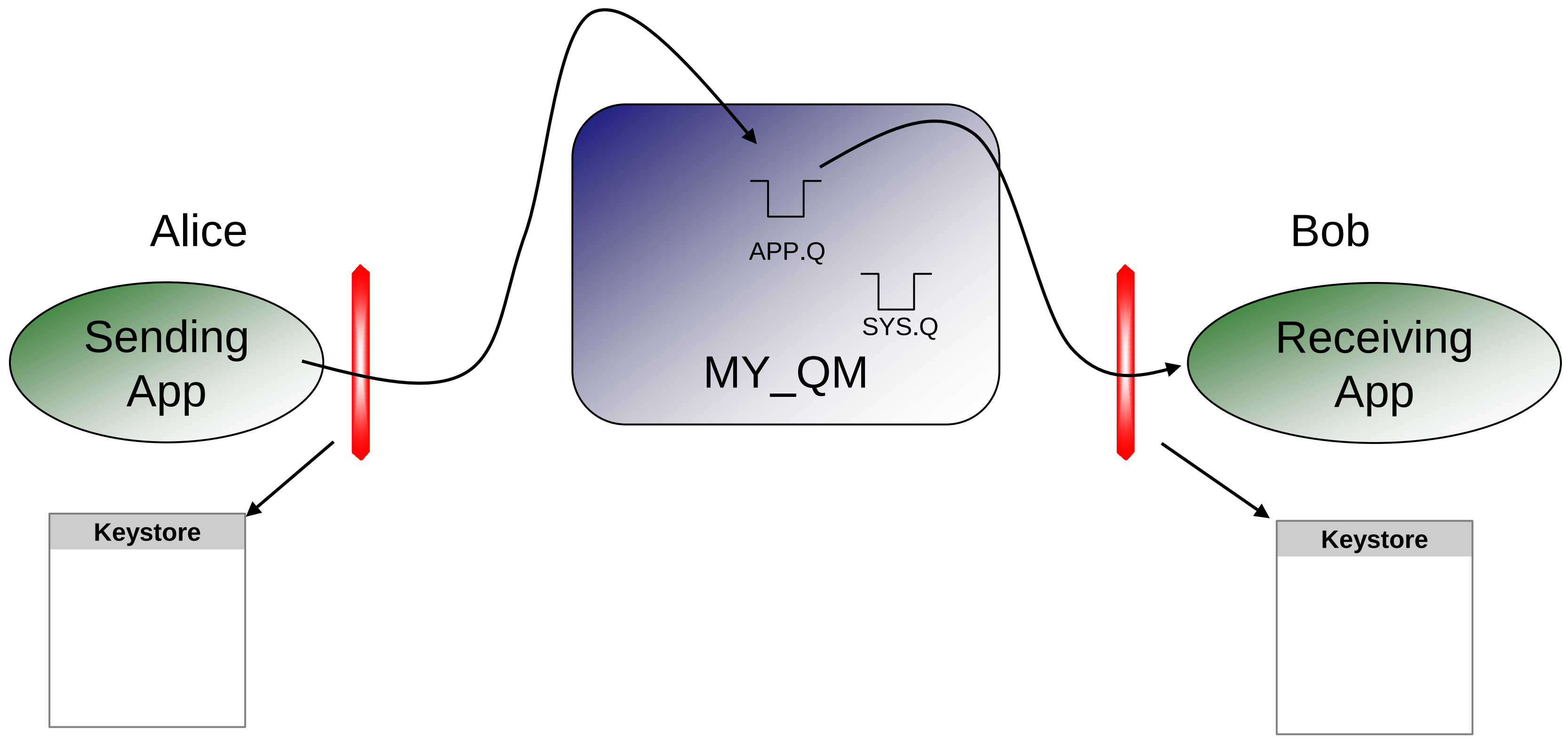
Available: 6,296,838,144 bytes

4

AMS z/OS installation

- SMP/E installation
  - Program 5655-W50, FMID HMS7701
  - Requires 120 tracks (Target), 150 tracks (Distribution), 1MB (zFS)
- Post-installation tasks
  - Update LPA for AMS modules in SDRQLINK
  - Update Authorized Program Facility (APF) list for SDRQLOAD
  - Update Program Properties Table (PPT) update
  - Update Product Registration list (IFAPRDxx)
  - Possible update to DIAG member for allocating in user storage key
- Create AMSM & AMSD procedures for the two AMS STCs
- Create profiles for STCs
  - Set up the userid(s) for the STCs, give SAF permissions,
- 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

WebSphere MQ AMS – install/config example, 2



1. Install AMS Interceptor
2. Configure AMS

AMS configuration

1. Enable AMS system queues

```
runmqsc MY_QM < "C:\WMQ AMS\bin\defineqs.mqs"
DEFINE QLOCAL(SYSTEM.PROTECTION.POLICY.QUEUE) MAXDEPTH(999999999)
   MAXMSGL(4194304) DEFSOPT(SHARED) SHARE DEFPSIST(YES)
DEFINE QLOCAL(SYSTEM.PROTECTION.ERROR.QUEUE) MAXDEPTH(999999999)
   MAXMSGL(4194304) DEFSOPT(SHARED) SHARE DEFPSIST(YES)
All valid MQSC commands were processed.
```

2. Activate AMS interceptors

```
cfgmqs -enable -server MY_QM

DRQDT3052I   The IBM WebSphere MQ Advanced Message Security server interceptor
  has been enabled successfully
```

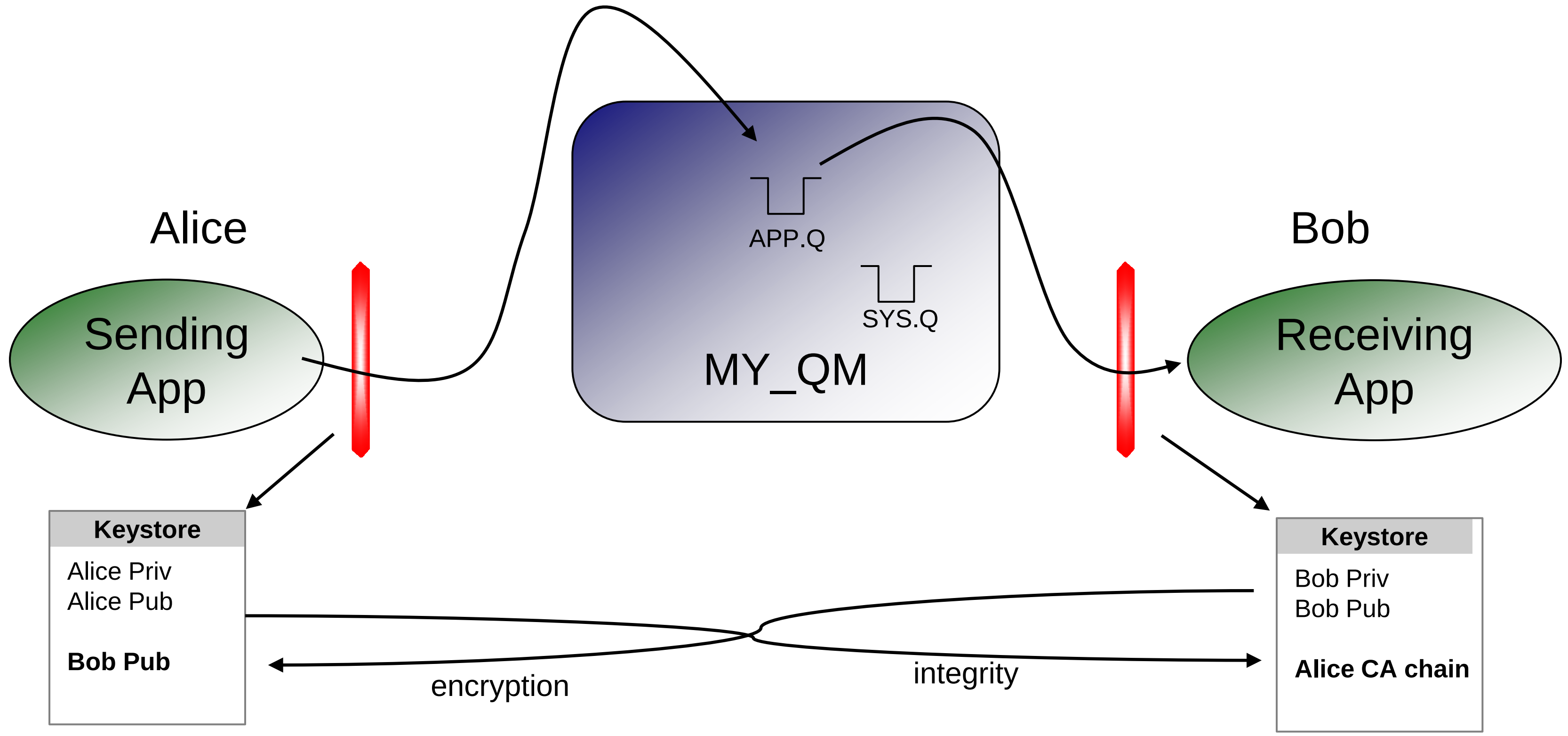
3. Set up Environment variable to point to AMS key database configuration

```
MQS_KEYSTORE_CONF=C:\AMSStuff\Carl\keystore.conf
```

4. Create the AMS key database configuration file, eg. `C:\AMSStuff\Carl\keystore.conf`

```
cms.keystore=C:/AMSStuff/Carl/carlkey
cms.certificate=Carl_Cert
```

WebSphere MQ AMS – install/config example, 3

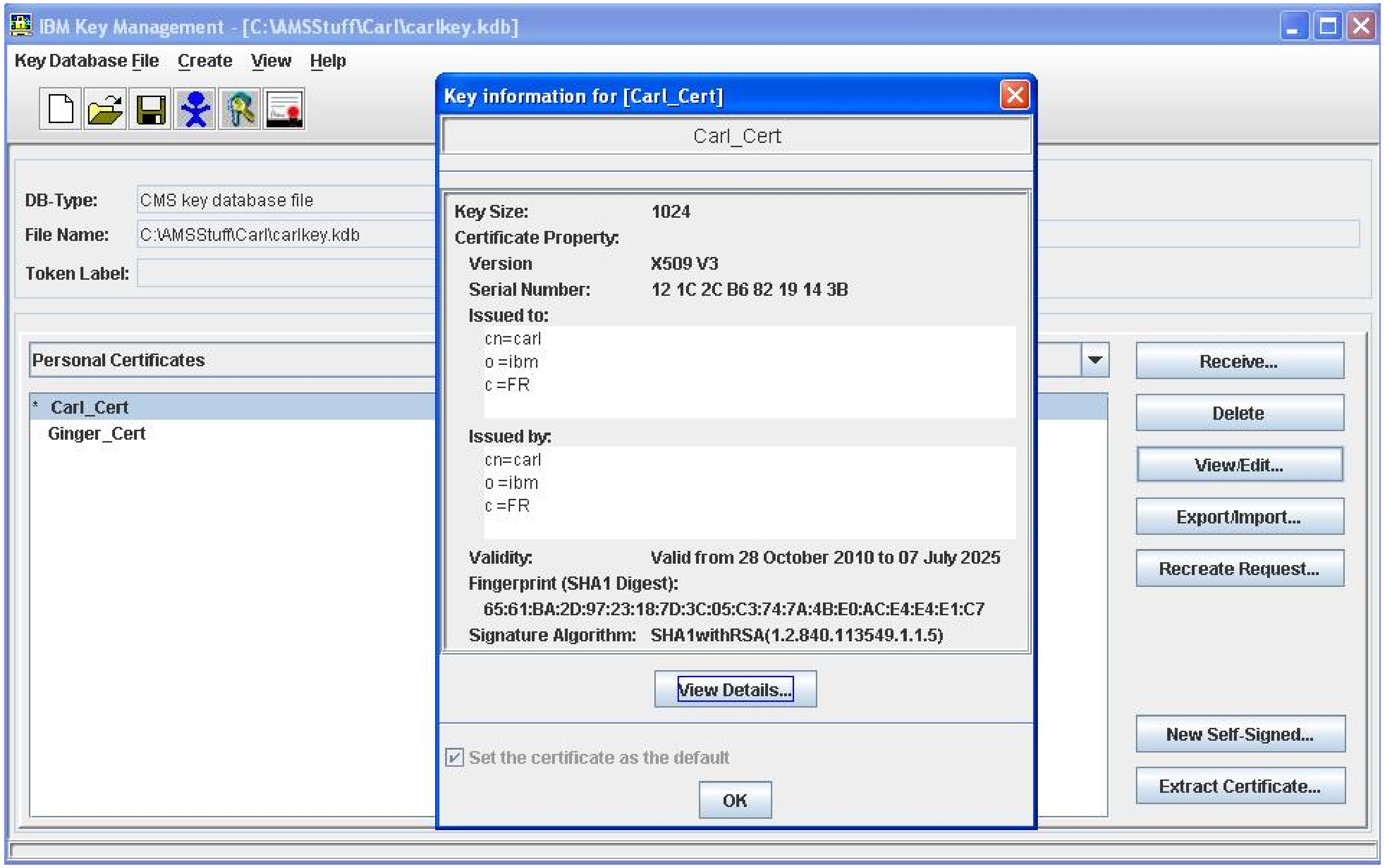


1. Install AMS Interceptor
2. Configure AMS
3. Create keystores with public / private key pairs
   a) copy sender's CA key chain to receiver's keystore for integrity
   b) copy receiver's public key to sender's keystore for encryption

2

Certificate management

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



DB-Type: CMS key database file
File Name: C:\AMSStuff\Carl\carlkey.kdb
Token Label:

Personal Certificates
* Carl_Cert
Ginger_Cert

Key information for [Carl_Cert]

Carl_Cert

```
Key Size:               1024
Certificate Property:
  Version               X509 V3
  Serial Number:        12 1C 2C B6 82 19 14 3B
  Issued to:
    cn=carl
    o =ibm
    c =FR
  Issued by:
    cn=carl
    o =ibm
    c =FR
  Validity:             Valid from 28 October 2010 to 07 July 2025
  Fingerprint (SHA1 Digest):
    65:61:BA:2D:97:23:18:7D:3C:05:C3:74:7A:4B:E0:AC:E4:E4:E1:C7
  Signature Algorithm:  SHA1withRSA(1.2.840.113549.1.1.5)
```

[✓] Set the certificate as the default

RACF on z/OS (sender side)

```
//SYSTSIN  DD *

  RACDCERT ID(QZ09AMS) ADDRING(drq.ams.keyring)

 /* Create a CA certificate good thru my retirement :> */
 RACDCERT CERTAUTH GENCERT -
   SUBJECTSDN( CN('AMS CertAuth') O('IBM') ) WITHLABEL('AMSCA') -
   KEYUSAGE(CERTSIGN) TRUST NOTAFTER(DATE(2018/12/31) )

 /* Connect the CA certificate to the AMSD Data task STC */
 RACDCERT ID(QZ09AMS) CONNECT(CERTAUTH LABEL('AMSCA') -
    RING(drq.ams.keyring)

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

 /* Make a keyring for a userid that will be a MQPUTer or MQGETer */
 RACDCERT ID(FARKAS) ADDRING(drq.ams.keyring)

 /* Create a Certificate for a MQPUTer or MQGETer id */
 RACDCERT ID(FARKAS) GENCERT -
  SUBJECTSDN( c('FR') O('IBM France') CN('Carl on Z') ) -
  WITHLABEL('CARLONZ') SIGNWITH(CERTAUTH LABEL('AMSCA') )  -
  NOTAFTER(DATE(2018/12/31) ) -
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)

   RACDCERT ID(FARKAS) CONNECT(ID(FARKAS) LABEL('CARLONZ') -
  RING(drq.ams.keyring) DEFAULT USAGE(PERSONAL))

   RACDCERT ID(QZ09AMS) ADD('FARKAS.CARLSS.CERT') TRUST -
     WITHLABEL('CARLSS')
   RACDCERT ID(QZ09AMS) CONNECT( ID(QZ09AMS) LABEL('CARLSS') -
     RING(drq.ams.keyring) USAGE(SITE) )

  SETROPTS RACLIST(FACILITY) REFRESH
/*
```

Data task on sender uses CA to validate

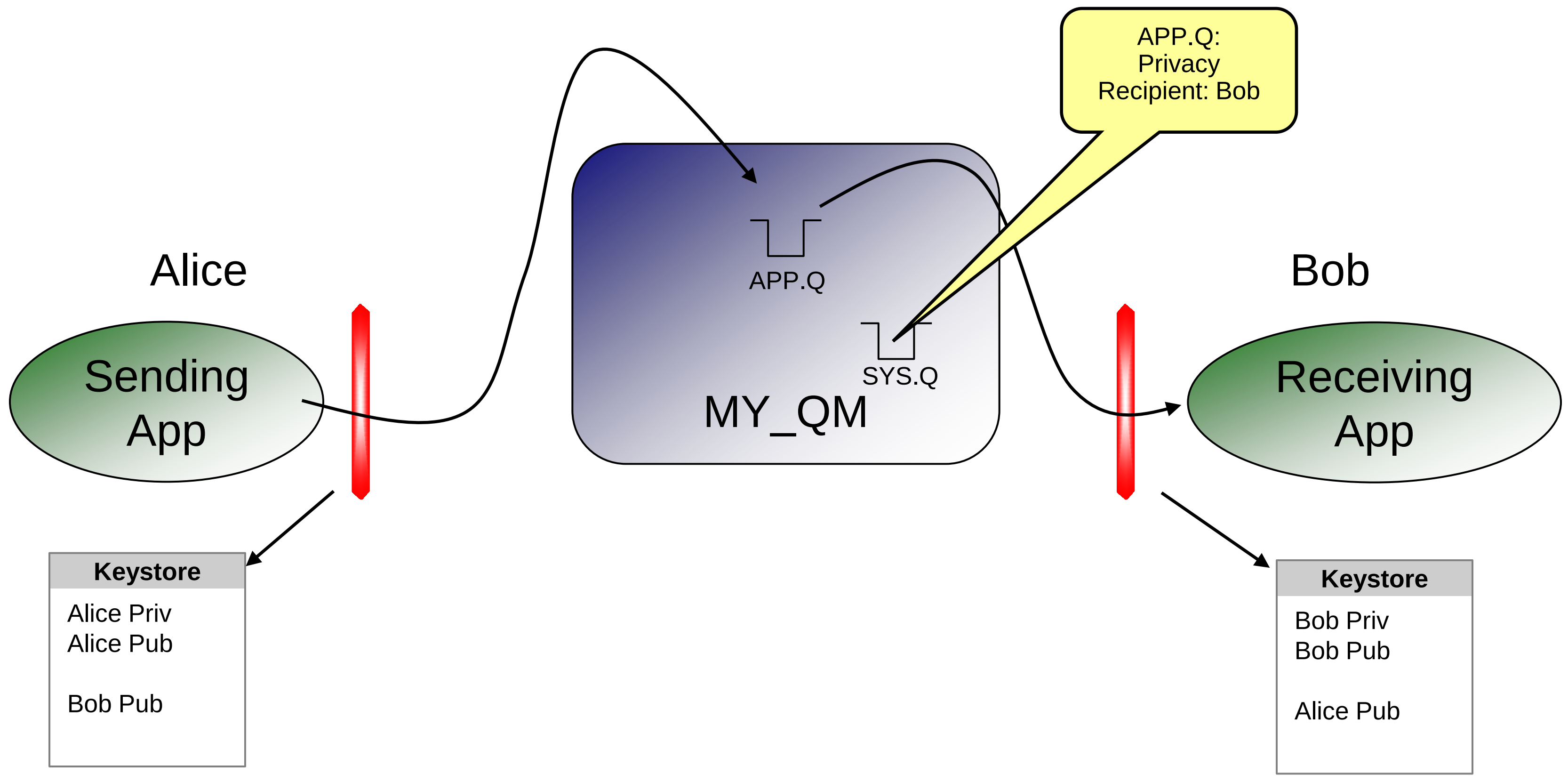
Export this CA so others can validate my certificate

PUTer userid on sender uses certif to sign

Import the Public certif of the Receiver so I can encrypt using this

3

WebSphere MQ AMS – install/config example, 4

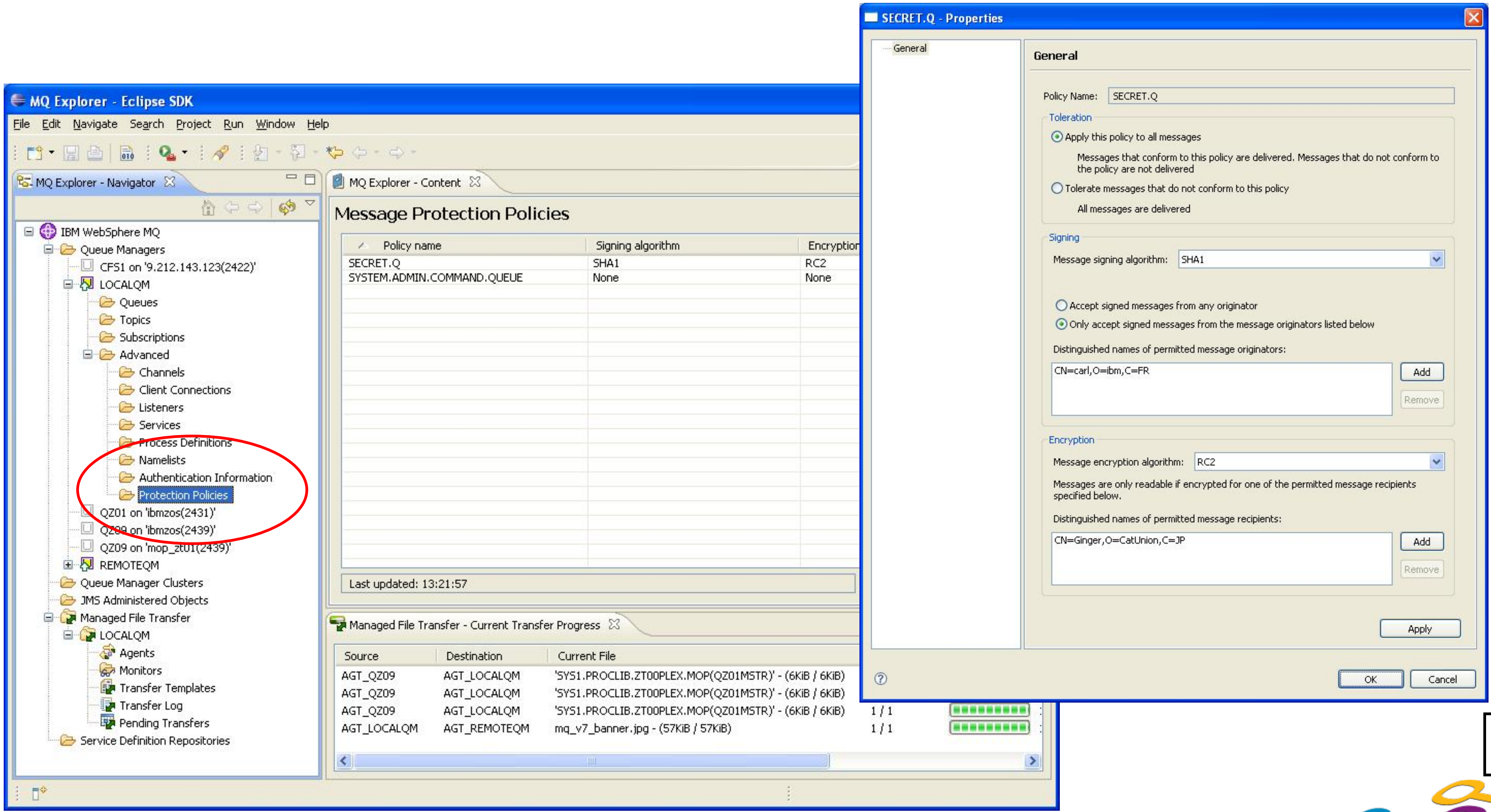


1. Install AMS Interceptor
2. Configure AMS
3. Setup keystores with public / private key pairs
4. Define protection policy for the queue (setmqspl)

Queue policy definition

- **Use either GUI or line-mode**

```
setmqspl -m LOCALQM -p SECRET.Q -s SHA1 -a
  "CN=carl,O=ibm,C=FR" -e RC2 –r “CN=Ginger,O=CatUnion,C=JP”
```



MQ Explorer - Eclipse SDK

File Edit Navigate Search Project Run Window Help

MQ Explorer - Navigator

- IBM WebSphere MQ
  - Queue Managers
    - CFS1 on '9.212.143.123(2422)'
    - LOCALQM
      - Queues
      - Topics
      - Subscriptions
      - Advanced
        - Channels
        - Client Connections
        - Listeners
        - Services
        - Process Definitions
        - Namelists
        - Authentication Information
        - Protection Policies
    - QZ01 on 'ibmzos(2431)'
    - QZ09 on 'ibmzos(2439)'
    - QZ09 on 'mop_zt01(2439)'
    - REMOTEQM
  - Queue Manager Clusters
  - JMS Administered Objects
  - Managed File Transfer
    - LOCALQM
      - Agents
      - Monitors
      - Transfer Templates
      - Transfer Log
      - Pending Transfers
  - Service Definition Repositories

MQ Explorer - Content

Message Protection Policies

Policy name	Signing algorithm	Encryption
SECRET.Q	SHA1	RC2
SYSTEM.ADMIN.COMMAND.QUEUE	None	None

Last updated: 13:21:57

Managed File Transfer - Current Transfer Progress

Source	Destination	Current File		
AGT_QZ09	AGT_LOCALQM	'SYS1.PROCLIB.ZTOOPLEX.MOP(QZ01MSTR)' - (6KiB / 6KiB)		
AGT_QZ09	AGT_LOCALQM	'SYS1.PROCLIB.ZTOOPLEX.MOP(QZ01MSTR)' - (6KiB / 6KiB)		
AGT_QZ09	AGT_LOCALQM	'SYS1.PROCLIB.ZTOOPLEX.MOP(QZ01MSTR)' - (6KiB / 6KiB)	1 / 1	🟩
AGT_LOCALQM	AGT_REMOTEQM	mq_v7_banner.jpg - (57KiB / 57KiB)	1 / 1	🟩

SECRET.Q - Properties

General

Policy Name: SECRET.Q

Toleration

- [✓ Apply this policy to all messages]
  Messages that conform to this policy are delivered. Messages that do not conform to the policy are not delivered
- Tolerate messages that do not conform to this policy
  All messages are delivered

Signing

Message signing algorithm: SHA1

- Accept signed messages from any originator
- [✓ Only accept signed messages from the message originators listed below]

Distinguished names of permitted message originators:

CN=carl,O=ibm,C=FR

Encryption

Message encryption algorithm: RC2

Messages are only readable if encrypted for one of the permitted message recipients specified below.

Distinguished names of permitted message recipients:

CN=Ginger,O=CatUnion,C=JP

1

AMS DRQUTIL commands on z/OS

```
//CFAMSAD JOB 'Make MQ AMS queues',CLASS=A,MSGLEVEL=(1,1),
// NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZT01
//*******************************************************************
//*    Administer MQ Advanced Message Service (AMS)                  *
//*******************************************************************
//          SET DIR='/u/farkas'
//          SET FN='drqdserv.envars'
//*
//DRQUTIL  EXEC PGM=DRQUTIL,
//          PARM='ENVAR("_CEE_ENVFILE=&DIR./&FN") /'
//STEPLIB  DD DSN=WMQ.AMS.V7R1.SDRQLOAD,DISP=SHR
//          DD DSN=WMQ.V7R0M1.SCSQANLE,DISP=SHR
//          DD DSN=WMQ.V7R0M1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 setmqspl -m QZ09
   -p TO.SECRET.FROMZ
   -s SHA1
   -e RC2 -r "CN=carl,O=ibm,C=FR"
/*
//
```

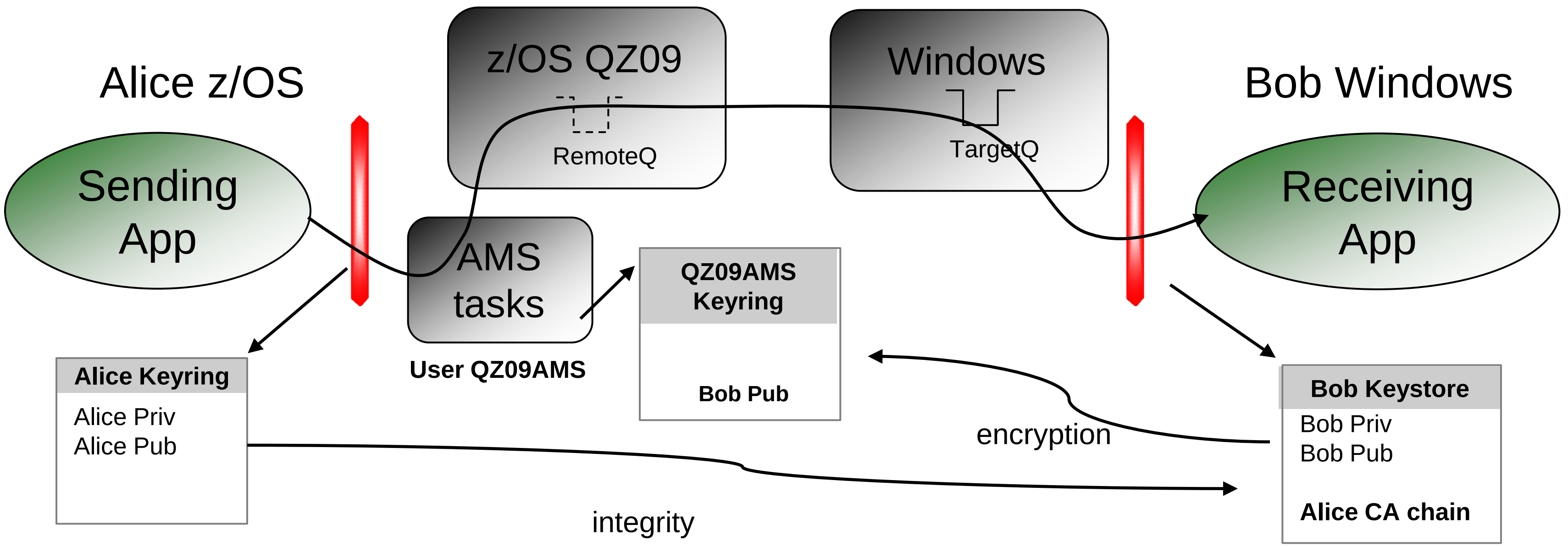
Point to parameters

Execute AMS admin commands

drqdserv.envars

```
 _DRQSERV_QMGR=QZ09
 _DRQSERV_MSG_LOGGING=stderr_logging
 _DRQSERV_MSG_LEVEL=*.i
 _DRQSERV_MSG_FOLDING=no
 _DRQ_INIT_THREADS=20
 _DRQ_MAX_THREADS=100
 NLSPATH=/usr/lpp/mqmese/V7R0M1/lib/nls/msg/%L/…£LANG=En_US.IBM-1047
 TZ=EST5EDT
```

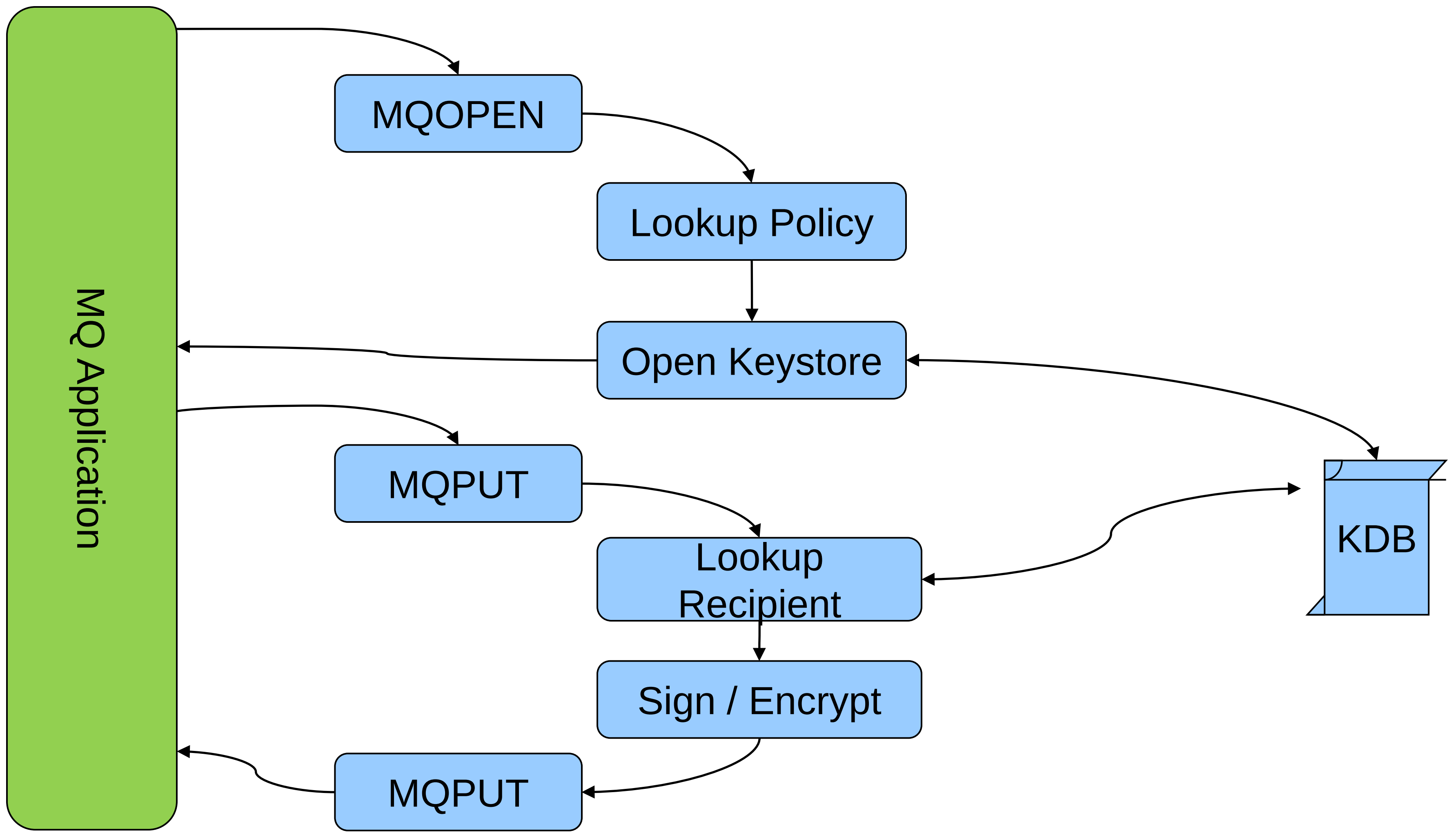
WebSphere MQ AMS – config z/OS -> Windows



1. Install AMS Interceptor
2. Configure AMS
3. Create keystores/keyrings with public / private key pairs
   a) copy sender's CA key chain to receiver's keystore for integrity
   b) copy receiver's public key to sender's keyring for encryption

2

MQAMS Interceptor Process



Error handling

- AMS returns a RC=2063 if the application tries to access (MQGET) a message for which it is not authorized

```
c:\result>amqsgbr SECRET.Q LOCALQM
Sample AMQSGBR0 (browse) start
LOCALQM
MQGET ended with reason code 2063
Sample AMQSGBR0 (browse) end
```

- The event is also logged in the <AMS installation>\log\*.log file

- 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.

Encrypted message examples

Encrypted message (via Q alias)

```
c:\result>amqsbcg SECRET.Q.ALIAS LOCALQM

AMQSBCG0 - starts here
**********************

 MQOPEN - 'SECRET.Q.ALIAS'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 437
  Format : '        '
  Priority : 0  Persistence : 0
  MsgId : X'414D51204C4F43414C514D2020202020D403CF4C201A0402'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'LOCALQM                                         '
  ** Identity Context
    :
****   Message      ****

 length - 1242 bytes

00000000:  5044 4D51 0200 0200 6800 0000 6800 0000 'PDMQ....h...h...'
00000010:  0800 0000 B501 0000 1100 0000 0000 0000 '................'
00000020:  4D51 5354 5220 2020 0000 0000 0000 0000 'MQSTR   ........'
00000030:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
00000040:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
00000050:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
00000060:  0000 0000 0000 0000 3082 046E 0609 2A86 '........0é.n.   ʻ
00000070:  4886 F70D 0107 03A0 8204 5F30 8204 5B02 'Hå.....áé._0é.[.'
00000080:  0100 3181 D730 81D4 0201 0030 3D30 3131 '..1ü.0ü....0=011'
00000090:  0B30 0906 0355 0406 1302 4A50 3111 300F '.0  ..U....JP1.0.
000000A0:  0603 5504 0A13 0843 6174 556E 696F 6E31 '..U....CatUnion1'
000000B0:  0F30 0D06 0355 0403 1306 4769 6E67 6572 '.0...U....Ginger'
000000C0:  0208 C1C7 B970 3999 57D6 300D 0609 2A86 '.....p9ÖW.0..
000000D0:  4886 F70D 0101 0105 0004 8180 649E 822A 'Hå........üÇd.é*'
000000E0:  C090 A27B 16BE E9BD 916C 8F50 C239 5B9E '.Éó{..Θ.ælÅP.9[.'
000000F0:  5C87 4F22 2A9F 0839 6B9D C11C 27B9 53D3 '\çO"*ƒ.9k¥..'.S.'
00000100:  2AC3 C929 B5D8 FB71 4D2B 8F39 A8B2 381D '*..)...qM+Å9¿.8.'
00000110:  31C8 C29D 7608 0891 D6B8 744B 8012 A9DF '1..¥v..æ..tKÇ...'
```

DLQ of encrypted message (via Q alias)

```
c:\result>amqsbcg SYSTEM.PROTECTION.ERROR.QUEUE LOCALQM

AMQSBCG0 - starts here
**********************

 MQOPEN - 'SYSTEM.PROTECTION.ERROR.QUEUE'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 437
    :
****   Message      ****

 length - 1398 bytes

00000000:  444C 4820 0100 0000 0F08 0000 5345 4352 'DLH ........SECR'
00000010:  4554 2E51 0000 0000 0000 0000 0000 0000 'ET.Q............'
00000020:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
00000030:  0000 0000 0000 0000 0000 0000 4C4F 4341 '............LOCA'
00000040:  4C51 4D20 2020 2020 2020 2020 2020 2020 'LQM             '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020 2020 2020 2202 0000 '            "...'
00000070:  B501 0000 2020 2020 2020 2020 0B00 0000 '....        ....'
00000080:  433A 5C57 4D51 5637 5C62 696E 5C61 6D71 'C:\WMQV7\bin\amq'
00000090:  7370 7574 2E65 7865 2020 2020 3230 3130 'sput.exe    2010'
000000A0:  3131 3135 3136 3236 3533 3530 5044 4D51 '111516265350PDMQ'
000000B0:  0200 0200 6800 0000 6800 0000 0800 0000 '....h...h.......'
000000C0:  B501 0000 0600 0000 0000 0000 4D51 5354 '............MQST'
000000D0:  5220 2020 0000 0000 0000 0000 0000 0000 'R   ............'
000000E0:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
000000F0:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
00000100:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
00000110:  0000 0000 3082 045E 0609 2A86 4886 F70D '....0é.^.   *åHå..'
00000120:  0107 03A0 8204 4F30 8204 4B02 0100 3181 '...áé.O0é.K...1ü'
00000130:  D730 81D4 0201 0030 3D30 3131 0B30 0906 '.0ü....0=011.0
00000140:  0355 0406 1302 4A50 3111 300F 0603 5504 '.U....JP1.0...U.'
00000150:  0A13 0843 6174 556E 696F 6E31 0F30 0D06 '...CatUnion1.0..'
00000160:  0355 0403 1306 4769 6E67 6572 0208 C1C7 '.U....Ginger....'
00000170:  B970 3999 57D6 300D 0609 2A86 4886 F70D '.p9ÖW.0..   *åHå..'
```

Why use X.509 certificates?

- WMQ AMS is based on asymmetric cryptography, also known as public key cryptography, using X.509 certificates.
- Common objections to certificate-based crypto include:
  - "Setting a pass-phrase in the config file makes admin tasks easy"
  - "But symmetric encryption is so much faster"
  - "I just need encryption at rest"
  - Remember these, we'll come back to them shortly.
- But these assume that the two options are otherwise equivalent
  - Or else assume that any loss of security will be minimal and offset by the added convenience
- The next few slides will examine why that might not be true

Why was public key crypto invented?

- Secure key exchange turns out to be a non-trivial problem
- If I can tell you the key over the phone, it is easy to crack
  - "the quick brown fox jumped over the lazy dog"
  - Reduced character set
  - Low entropy
- If it's too complex to read to you, I now need to arrange for secure delivery
  - 5lCXc-)7DRY/0hv'cEdA50{T{z8$!',[lBAR^RP%#Xv_na(>ag:>Y@m*eGh]Ko]
  - Do you want to try to type this in correctly?
  - Could send you a flash drive by secure courier
    - But it's still low entropy
    - And if you cut-and-paste, it leaves traces in memory
- The list of issues with key distribution is rather extensive

Role of asymmetric keys to solve the key distribution problem

- One-way cryptographic algorithm based on dual keys
  - Anything encrypted with one key can only be decrypted with the other key from the same pair
  - One key designated as "private" and kept strictly confidential
  - One key designated as "public" and distributed freely
- Possible to perform a zero-knowledge key exchange
  - Dynamcally generate a unique, long session key
  - High entropy, largest possible key space (binary)
  - Public/private keys facilitate an exchange such that the session key cannot be intercepted, even while an attacker is watching
    - (See: Diffie-Hellman key exchange)

Additional benefits of asymmetric keys

- Authentication – A message that can be decrypted using the public key can only have been generated with the private key
  - Recipient has assurance of origin of the message
- Authorization – Encrypt a message using a public key and only the holder of the private key can read it
  - Sender has assurance that only intended recipient has access
- Accountability – A message signed with a given private key can not have originated from someone who does not hold that private key
  - Non-repudiation by sender of messages signed with private key

Symmetric key has none of these benefits

- Example scenario: Consider a server with many clients, all using a shared symmetric key
  - Server has no cryptographic basis with which to distinguish one client from another
    - Must trust the identity asserted in the message
  - Clients cannot tell if a given reply message actually came from the server was spoofed by another key holder
  - Messages may be altered by any key holder without detection
  - Any sender can deny having sent a particular message
    - Because there is no way to prove otherwise

Objection #1 debunked

"Setting a pass-phrase in the config file makes admin tasks easy."

- Significant effective loss of security due to:
  - Typeable keys easier to crack
  - Difficulty in exchanging long, complex keys securely
- No support for
  - Authentication
  - Authorization
  - Accountability

The convenience of symmetric key is real but the differential in the level security achieved is tremendous. In terms of key strength alone, this amounts to *hundreds of orders of magnitude.*

Consider how security systems degrade

- Does the system scale?
- How does the system respond when a single node is compromised?
- What is the impact of a key update?
- Does the addition or deletion of individual nodes affect overall security of the system?

Too often, security decisions focus on the cost to deploy and operate without considering the additional impacts of routine maintenance, revocation, compromise or recovery.

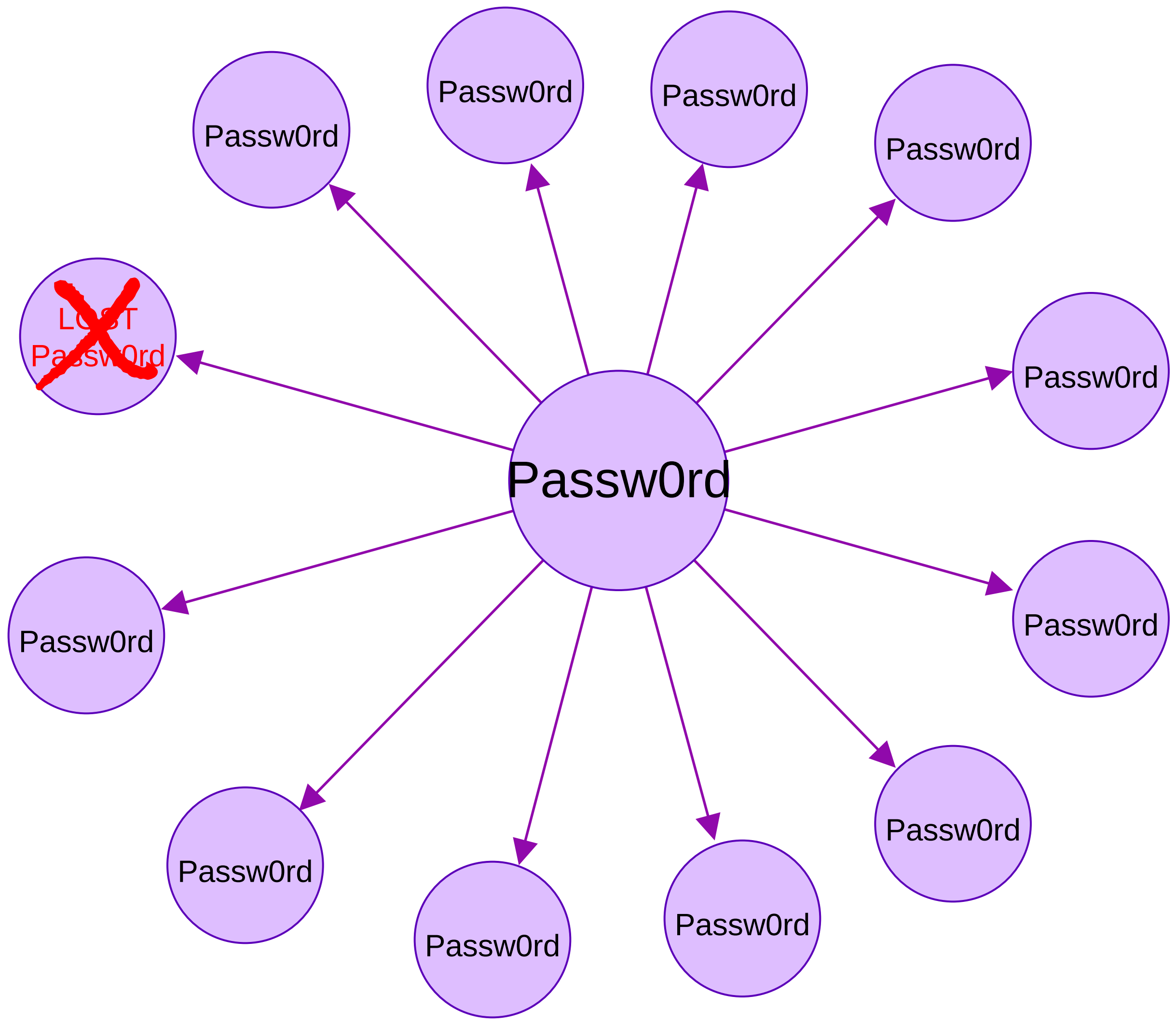
Symmetric key scaling

- Consider: A central hub and many remote locations
  - Each location must know the secret key
    - Dependent on physical security at each location
    - Dependent on at least one person per location
    - Ignore non-trivial key distribution problem for the moment
- What is the risk of a network with 10 spoke nodes?
- How does it increase when there are 100 spokes?
- How about 1,000 spokes?
- Actual customer use case was a corporate office and 3,200 locations

With symmetric key, the probability of compromise increases with each new node, eventually approaching certainty.

Symmetric key compromise

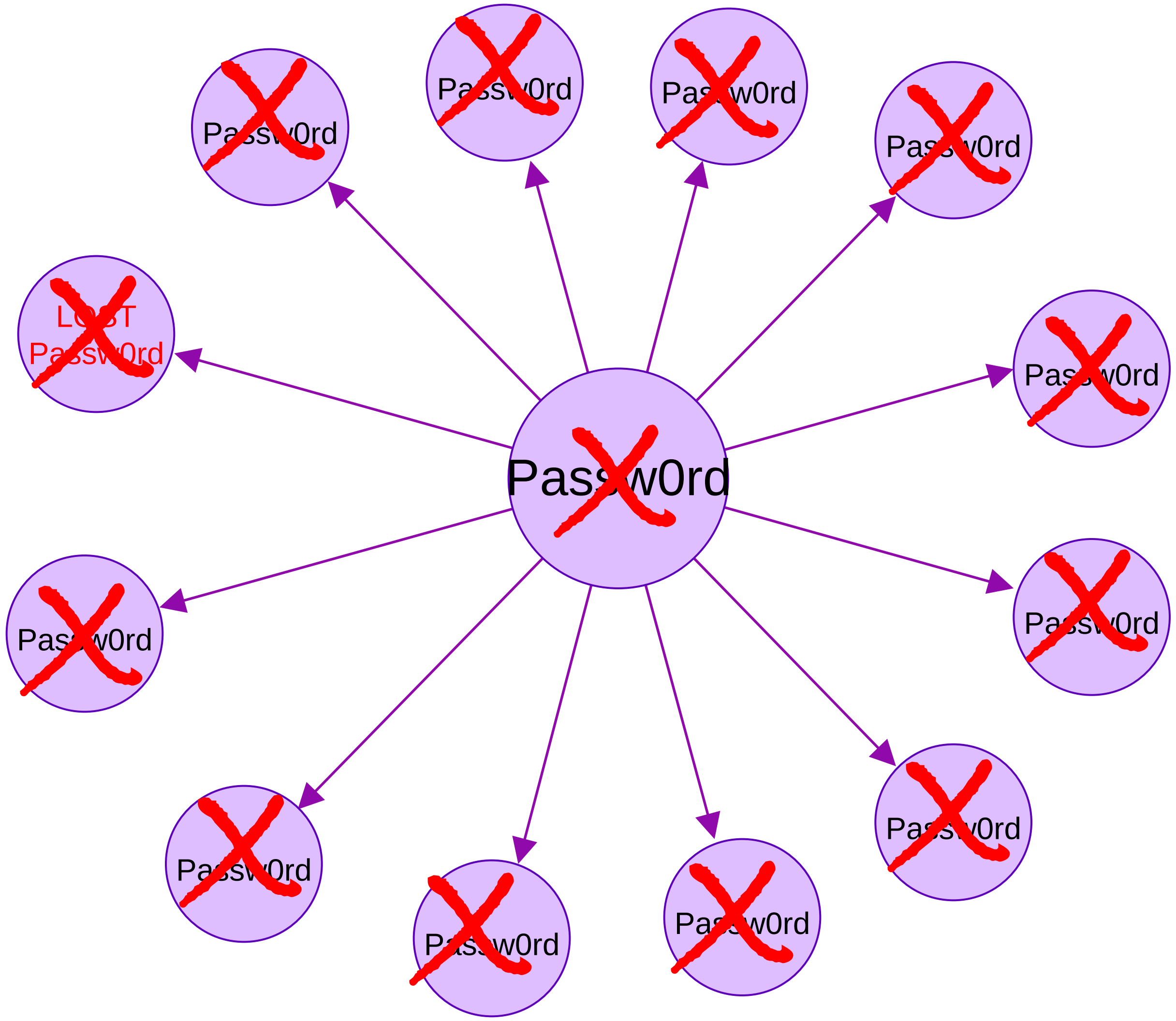
Compromise
 one
password…



Symmetric key compromise

Compromise
 one
password…

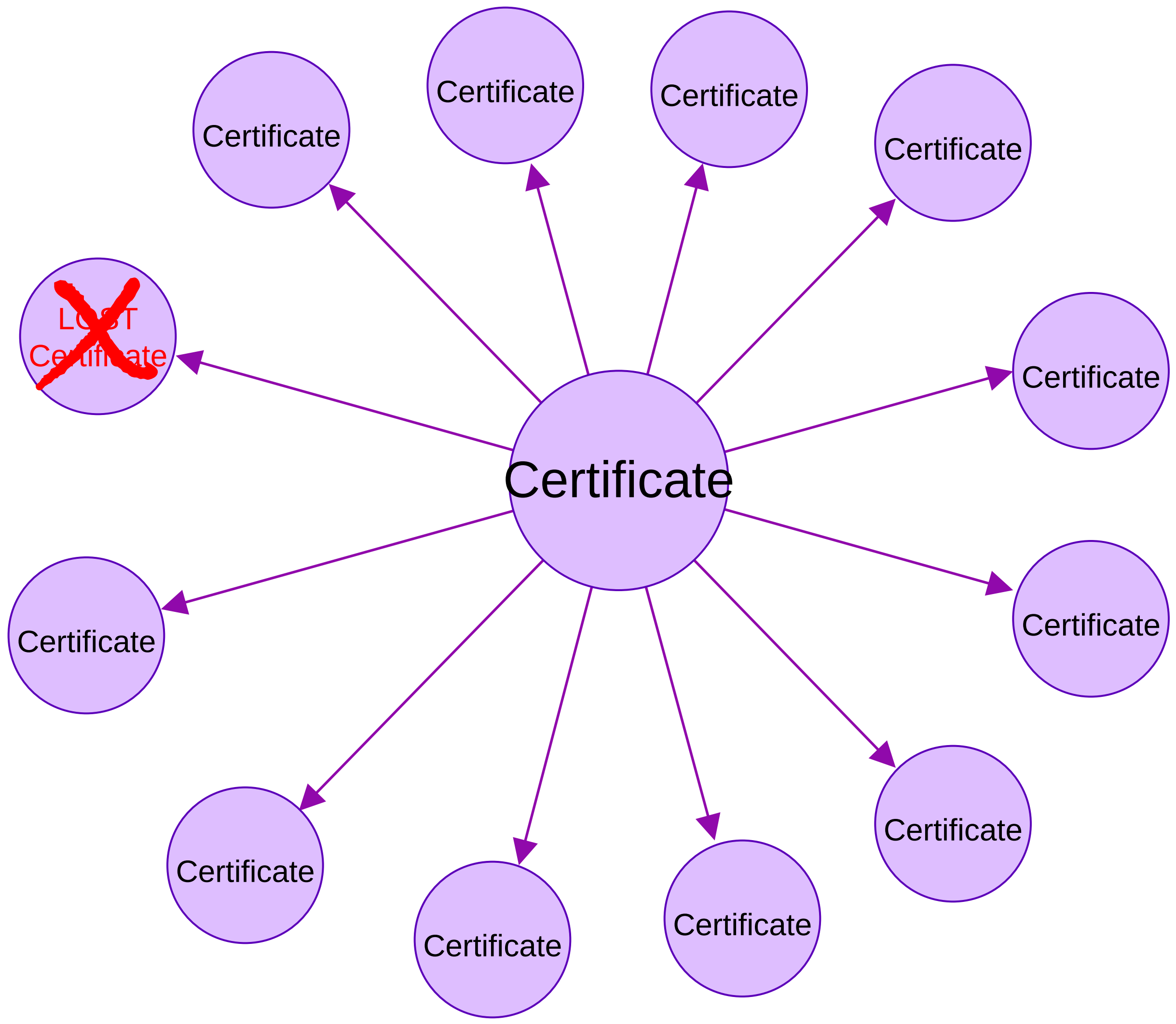
…and the
entire
network is
compromised.



Degrades catastrophically

Compare to asymmetric key compromise

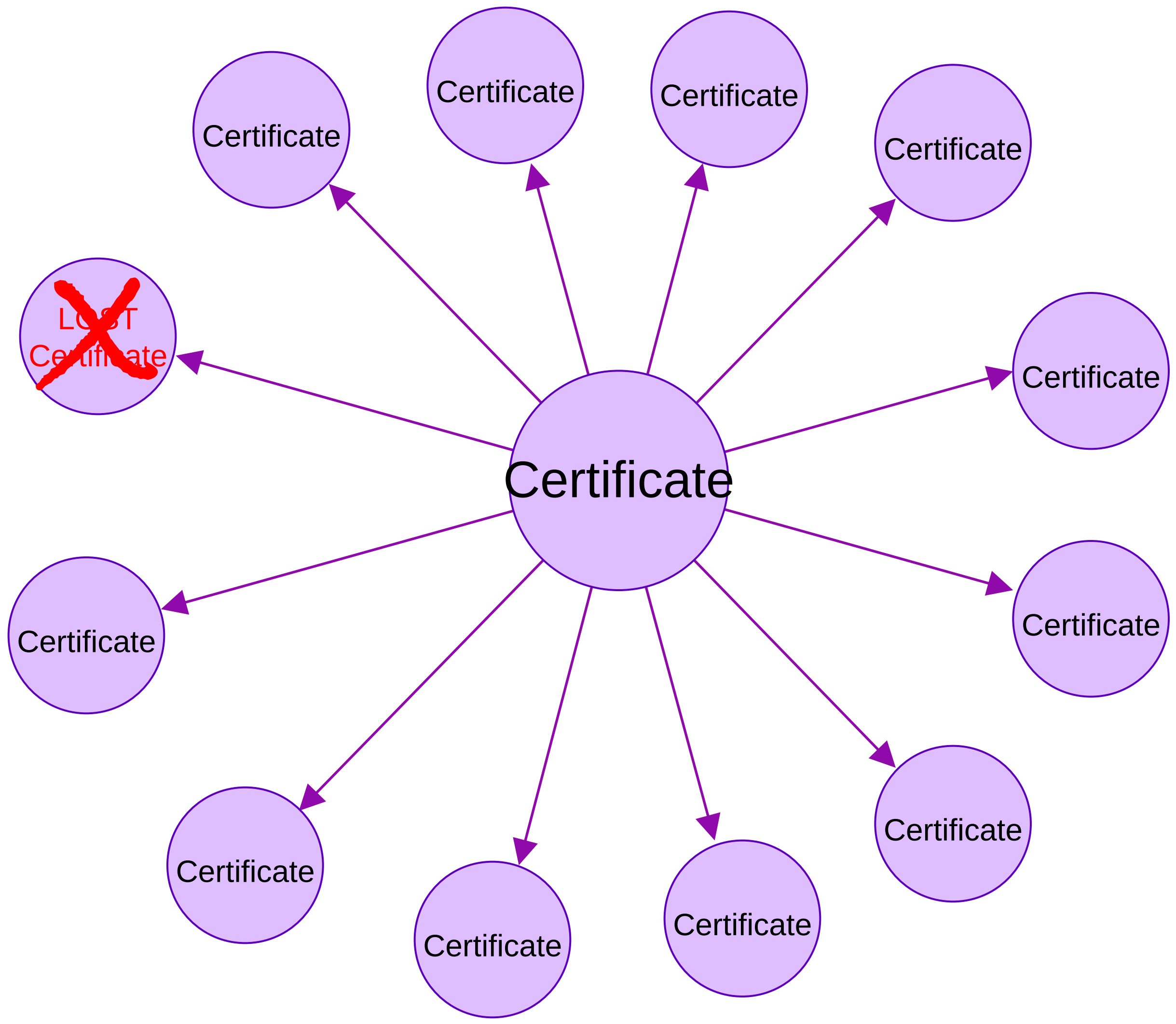
Compromise
one
certificate…



Compare to asymmetric key compromise

Compromise one certificate…

…only that node is compromised.



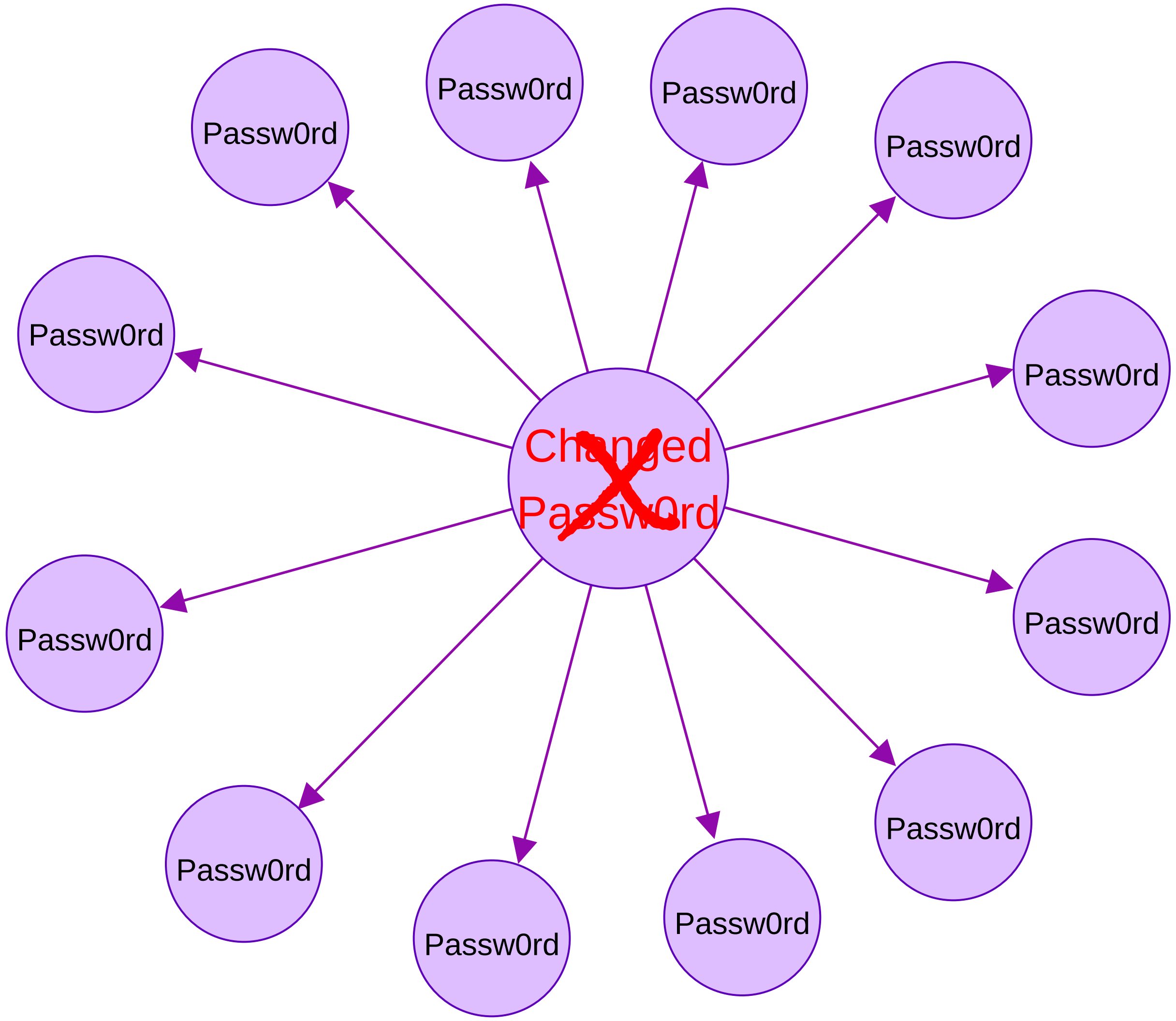
Degrades gracefully

What about key updates?

- Regulatory regimes typically mandate periodic key updates
- Certificates can be renewed (as opposed to replaced) with minimal impact
  - Update the keystore and bounce the app
- Even cert replacement causes only minimal disruption
  - No coordinated system-wide outage
  - Each instance is bounced on its own schedule
    - Add new certificates
    - Rolling bounce for instances to pick up new certificates
    - Update relevant policies to point to new cert
    - Remove old cert
    - Rolling bounce to pick up updated keystore

Symmetric key renewal

Change
one
password…

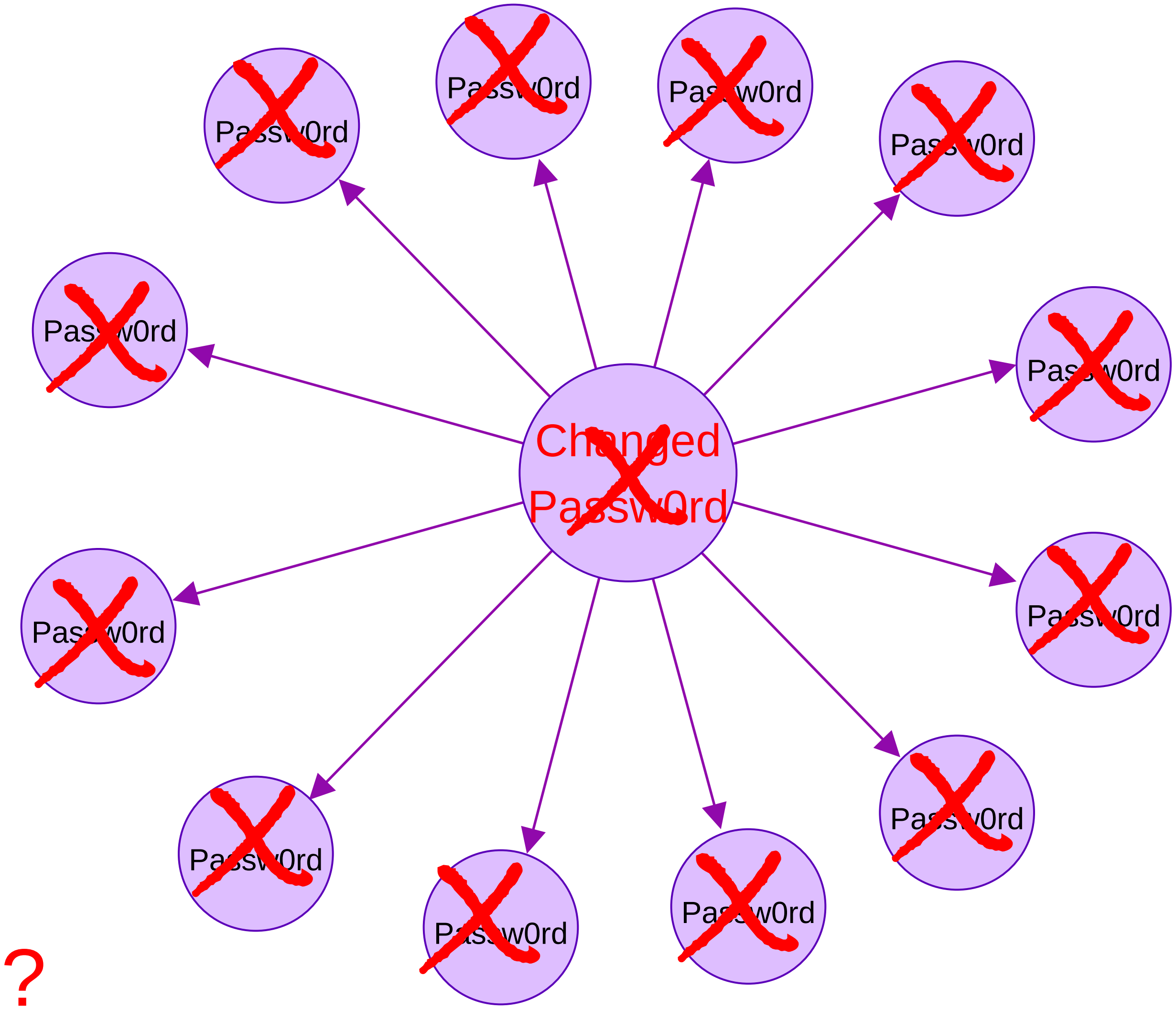


Symmetric key renewal

Change
 one
password…

…and the
entire
network is
down.

Look familiar?



Add or delete a node

- Adding nodes is deceptively simple with symmetric key
  - This is why it looks so attractive at first
    - Even so, few people stop to consider what it means for dozens or hundreds of people to know the secret key
    - Or the implications of depending on physical security for the *same* secret key across hundreds or thousands of locations
- But what happens when a node is removed?
  - Has the key been securely destroyed? How do you know?
    - If you're not willing to bet the business on it, then you'd better take an outage while you re-key the entire network
  - If not destroyed, the credentials on that server can be used to compromise the entire network
    - No way to distinguish the attacker from legitimate nodes
    - No way to authenticate the messages

Symmetric encryption offers no means with which
to revoke access for an individual node

Objection #2 debunked

"But symmetric encryption is so much faster."

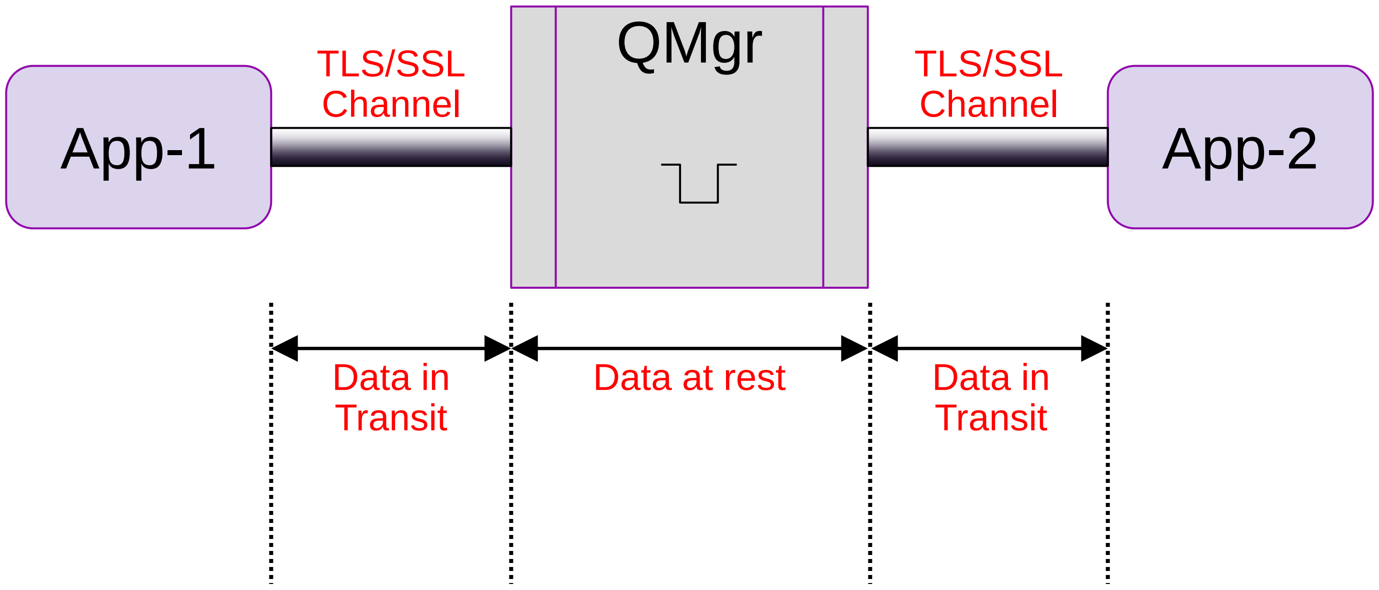
- Yes, it absolutely is.
  - But it doesn't scale
  - Routine maintenance requires a system-wide coordinated outage
  - It lacks any revocation mechanisms
  - And in the event of compromise, it degrades catastrophically

If this company is managing YOUR money, YOUR personal information or YOUR health data, do you want them using symmetric or asymmetric key encryption?

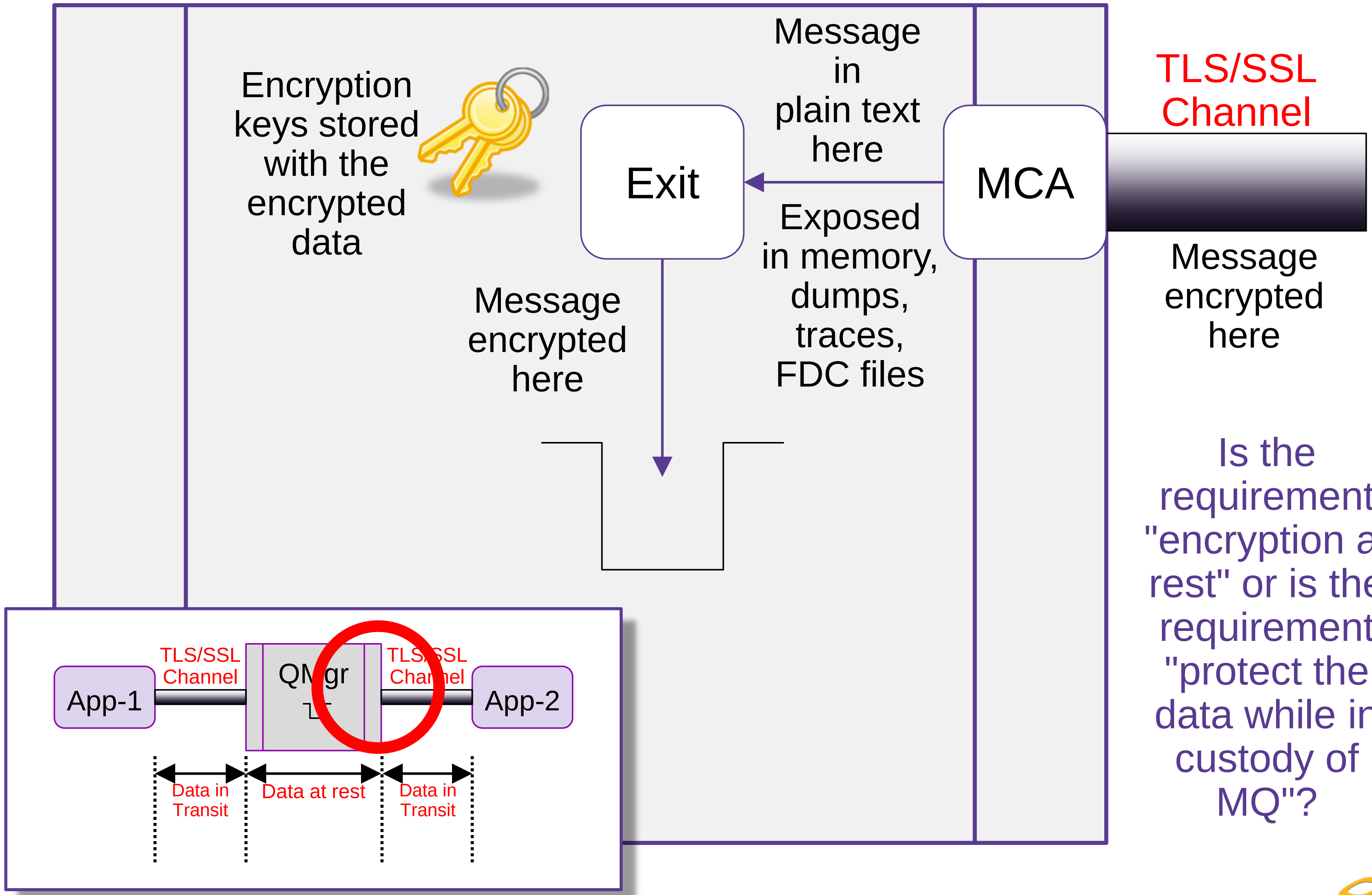
If it's your company, which encryption do you bet the business on?

Requirement vs. Implementation

- Is "encryption at rest" actually a requirement? Or is it just an implementation detail?
- The term is based on a gross oversimplification of the security model of messaging



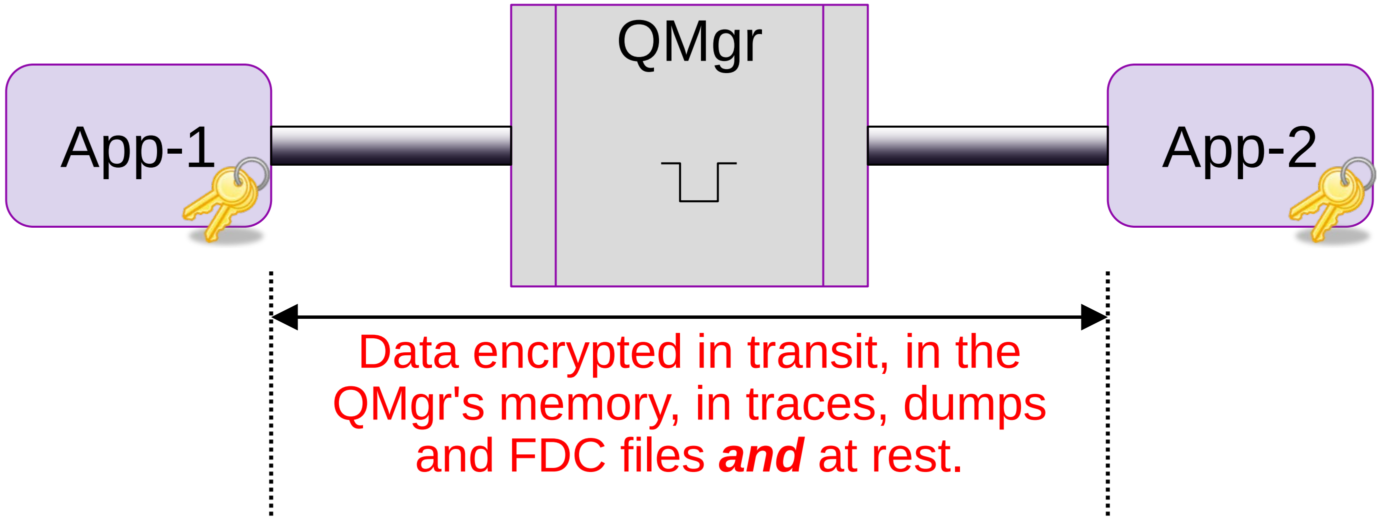
A closer look at the "at rest" phase



Is the requirement "encryption at rest" or is the requirement "protect the data while in custody of MQ"?

AMS provides true end-to-end

- AMS intercepts the message before it hits the channel
- Not in plaintext at any point until the receiver has it in memory
- Keys are held at the endpoints so compromise of the QMgr or its underlying server does not expose the data


QMgr
App-1
App-2
Data encrypted in transit, in the QMgr's memory, in traces, dumps and FDC files *and* at rest.

Note: AMS includes a server-side solution as in the previous slide, for when it is not possible to run AMS on the client application node. Use where appropriate but be aware of the difference in effective security with the different solutions.

Objection #3 debunked

"I just need encryption at rest."

- "Encryption at rest" does not describe a requirement
  - It is a consequence of a requirement to secure data while in custody of the QMgr but doesn't tell the whole story
- What is the actual threat to be mitigated?
  - That somebody steals the disk drive?
  - Or that the QMgr or server could be compromised?
- Perhaps the threat to be mitigated is failing the audit
  - Can you prove to the auditor that data cannot leak in memory, traces, dumps or log files?
  - Do you expect a savvy auditor to overlook such exposures?

Why use AMS section summary

- Public-key crypto wasn't invented because people were sitting around with nothing better to do
  - Driven by business requirements to overcome weaknesses inherent in the alternatives
- WebSphere MQ AMS uses public-key crypto not because it is "state of the art" or "the best security possible" but because it solves the key distribution problem and provides additional functionality not otherwise available such as:
  - Authentication
  - Authorization
  - Accountability
- Difference in performance can be great but it is linear
- Difference in effective security is exponential

Known limitations

- Covered in more detail on the following slides:
  - Pub/Sub
  - Channel data conversion
  - Distribution lists

- IMS Bridge not supported
- Non-threaded applications using API exit on HP-UX
- JMS and Java supported only with MQv7+
  - But v6.0 is EOS as of September 2012

Please submit and vote on enhancement requests in the RFE Community at http://ibm.co/IBMRFE

Pub/Sub

- The whole idea of pub/sub is to decouple publishers from subscribers and participation is dynamic
- If you don't know who your recipients are ahead of time, you cannot encrypt a message for them using their public key
  - Symmetric key crypto works for this use case but as we saw earlier the effective level of security is greatly reduced and such systems tend to degrade catastrophically when compromised
- But if you **do** know who your recipients are ahead of time, you can achieve the desired result using an alias over a topic.
- The use case of *signing* a publication is valid for pub/sub but not currently supported.
  - If you need this, please submit an RFE! http://ibm.co/IBMRFE

Channel data conversion

- This is not supported in AMS for a very simple reason – in order to perform data conversion, the data must be in plaintext *while in custody of the channel*.
  - That's why it's called *channel* data conversion ☺
  - If you can figure out how to perform code page conversion on encrypted data, get your affairs in order because some men in black will be stopping by for a visit and you won't be seen or heard from for a very long time
- Use convert on the GET call instead
  - May need to configure Java at some fix pack levels to convert at the client instead of the QMgr. See APAR IC72897 http://www-01.ibm.com/support/docview.wss?uid=swg21501765

API Distribution lists

- The API version of distribution list is not supported
- However an administrative distribution list may be configured using an alias over a topic
  - Publisher "thinks" it is putting messages on a queue
  - Recipients use an administrative subscription which delivers publications to a pre-defined queue
  - AMS policy is then defined for the alias and the recipient's queue

Summary

WebSphere MQ Advanced Message Security V7.0.1

- Protects message integrity and/or privacy
- Supports MQ v6 and v7.x
- Supports MQ Server, MQ Client and Java/JMS
- Existing MQ applications do not require changes
- “Light weight” product - No pre-requisites, easy installation, easy configuration
- 90-Day trial version available for download

Bibliography

- WMQ AMS InfoCenter at http://publib.boulder.ibm.com/infocenter/mqams/v7r0m1/index.jsp
- Program Directory for IBM WebSphere MQ Advanced Message Security for z/OS (GI13-0559)
- WebSphere MQ AMS Administration Guide (GC34-7142)
- Trial downloads http://www.ibm.com/developerworks/websphere/downloads/

MQ this week…

Mon 10:45 - 12:00	1577 - WebSphere MQ: Securing your Queue Manager*
Mon 14:00 - 15:15	1576 - Introduction to WebSphere MQ*
Mon 14:00 - 15:15	1597 - Roundtable: WebSphere MQ Feedback*
Mon 15:45 - 17:00	2255 - WebSphere Connectivity and Integration Feature Session with Q&A Panel
Mon 17:15 - 18:30	1575 - What's New in the WebSphere MQ Family of Products
Tue 10:45 - 12:00	1579 - WebSphere MQ for Managed File Transfer
Tue 10:45 - 12:00	1592 - WebSphere MQ: Machine 2 Machine Communications using Telemetry
Tue 13:30 - 14:45	1593 - WebSphere MQ: Publish/Subscribe Messaging
Tue 13:30 - 16:30	1595 - Hands-on Lab: WebSphere MQ
Tue 15:15 - 16:30	1581 - Extending WebSphere MQ and WebSphere Message Broker to the Cloud
Tue 15:15 - 16:30	1597 - Roundtable: WebSphere MQ Feedback*
Tue 16:45 - 18:00	1576 - Introduction to WebSphere MQ*
Wed 9:00 - 10:15	1589 - WebSphere MQ: What is your system up to?*
Wed 10:45 - 12:00	1578 - WebSphere MQ: Securing your Messages*
Wed 10:45 - 12:00	1591 - WebSphere MQ for zOS Internals
Wed 13:30 - 14:45	1596 - Meet the Experts: WebSphere MQ*
Wed 13:30 - 14:45	1585 - WebSphere MQ: Connecting to the Internet of Things
Wed 13:30 - 14:45	1586 - Using IBM WebSphere Application Server and IBM WebSphere MQ Together*
Wed 13:30 - 16:30	1594 - Hands-on Lab: WebSphere MQ Security (Distributed Platforms)
Wed 15:15 - 16:30	1584 - WebSphere MQ: Highly Scalable Publish Subscribe using Multicast
Wed 16:45 - 18:00	1588 - WebSphere MQ for Distributed Platforms Performance
Wed 16:45 - 18:00	1597 - Roundtable: WebSphere MQ Feedback*

[* Repeat Sessions]

and there's more…

MQ this week…

Thu 8:45 - 10:00	1590 - WebSphere MQ for Distributed Platforms Internals
Thu 8:45 - 10:00	1587 - WebSphere MQ for z/OS Performance
Thu 8:45 - 10:00	1597 - Roundtable: WebSphere MQ Feedback*
Thu 10:30 - 11:45	1596 - Meet the Experts: WebSphere MQ*
Thu 10:30 - 11:45	1586 - Using IBM WebSphere Application Server and IBM WebSphere MQ Together*
Thu 13:30 - 14:45	1589 - WebSphere MQ: What is your system up to?*
Thu 13:30 - 14:45	1580 - WebSphere MQ: Simplifying Migration
Thu 15:15 - 16:30	1582 - WebSphere MQ for z/OS Shared Queues (Advanced)
Thu 16:45 - 18:30	1583 - WebSphere MQ: Clustering update
Fri 08:45 - 10:00	1577 - WebSphere MQ: Securing your Queue Manager*
Fri 10:15 - 11:30	1578 - WebSphere MQ: Securing your Messages*

[* Repeat Sessions]

We love your Feedback!

- Don’t forget to submit your Impact session and speaker feedback! Your feedback is very important to us, we use it to improve our conference for you next year.
- Go to impactsmartsite.com from your mobile device
- From the Impact 2012 Online Conference Guide:
  - Select Agenda
  - Navigate to the session you want to give feedback on
  - Select the session or speaker feedback links
  - Submit your feedback

Copyright and Trademarks