WebSphere software

11:15 am

Introduction to SSL or, What's New in MQ Channel Security

Morag Hughson
hughson@uk.ibm.com

IBM Software Group

Channel Security - Agenda

- Security Problems
- Secure Sockets Layer (SSL)
- WebSphere MQ and SSL
  - WebSphere MQ Configuration Tasks
  - Security Administration Tasks

Security Problems

- **Eavesdropping**
  - **How do I stop someone from seeing the information I send?**
- **Tampering**
  - **How can I detect if someone has intercepted my information and changed it?**
- **Impersonation**
  - **How can I be sure who the information is from?**
  - **How can I be sure who I am exchanging information with?**

Security Problems : Eavesdropping - Notes

N O T E S

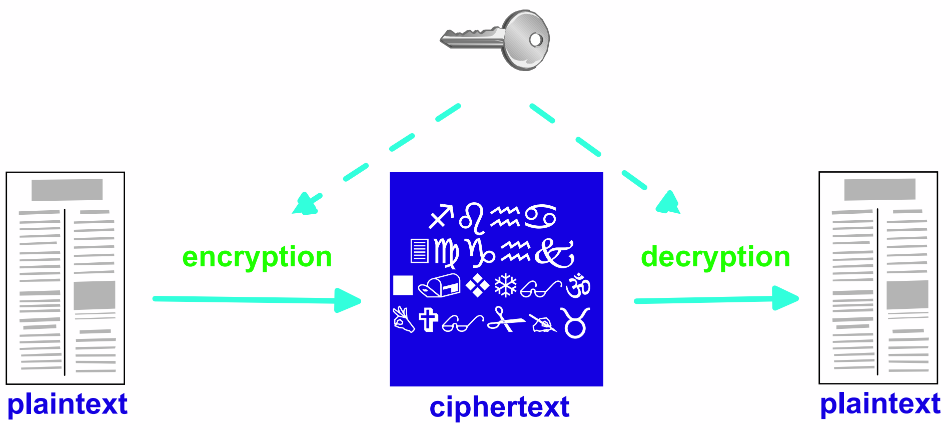
We are going to look at three main security problems in turn, eavesdropping, tampering and impersonation.

To deal with the problem of eavesdropping, you need to encrypt your information before you send it so that an eavesdropper cannot read your information. We will look at the different ways you can encrypt your message and what the problems are associated with them.

To deal with the problem of tampering, you can use a one-way hash function.

To deal with the problem of impersonation, you need to be able to verify the sender of the information and also to be able to authenticate the partner you are exchanging information with. Only the owner of the public key can decrypt information encrypted using it. However, how do you really know who the owner of the public key is?

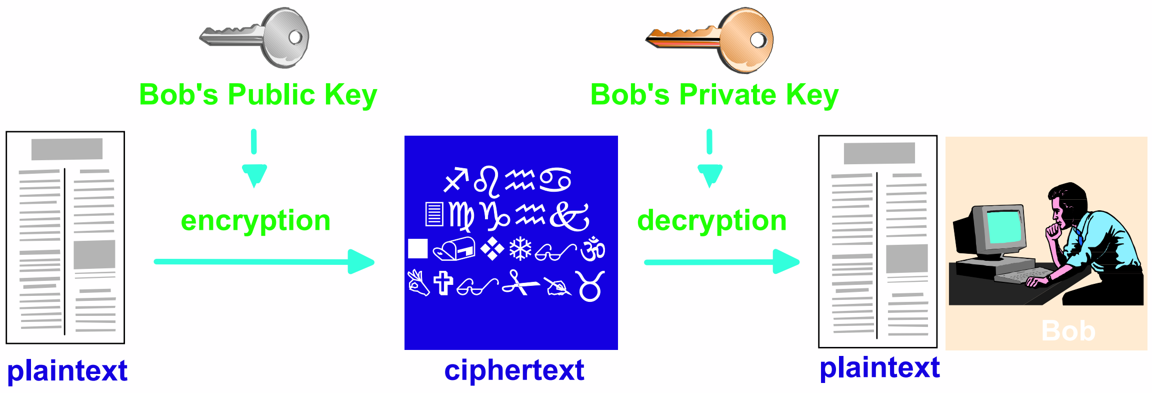
Cryptography



Symmetric Key

Secret Key

Cryptography



Asymmetric Key

Public/Private Key Pairs

Cryptography - Notes

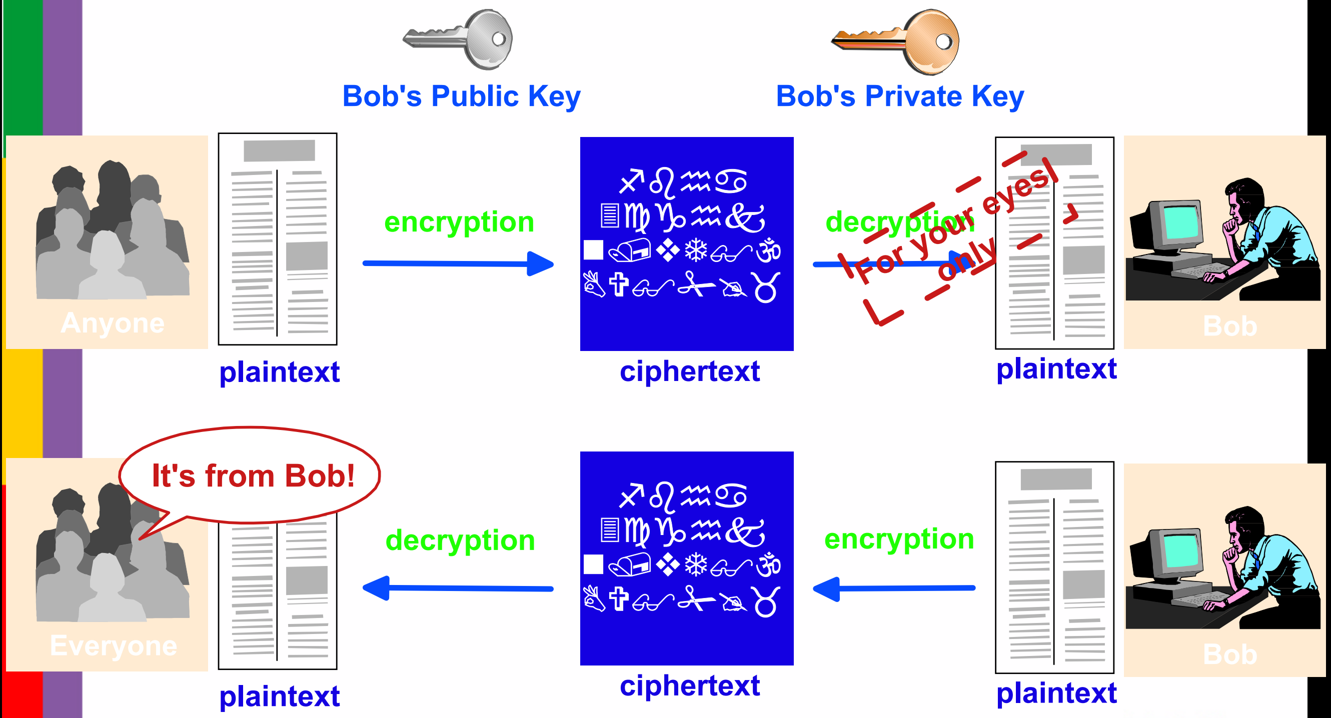
N O T E S

The eavesdropping problem can be solved by encrypting your information. We will look at symmetric and asymmetric keys.

The sender and the recipient could secretly agree on an algorithm to use to encrypt your information, a very simple example of which is add 'n' to each byte of your information, and the recipient subtracts 'n' from your encrypted information to get back to the original. This is known as a symmetric key algorithm - you can reverse the algorithm to decrypt the information. This type of key is also known as a secret key. This kind of key does however pose one problem. How do you secretly agree the algorithm? This is known as the key distribution problem.

The other way is that you publish an algorithm for everyone to use to encrypt information to send you and then use a private algorithm to decrypt the information. The public algorithm cannot be reverse engineered to create the private algorithm. This is known as an asymmetric key algorithm. In this case there is a public key known to all, and a private key known only to you. There is no key distribution problem here.

Asymmetric Key - Use both ways



Asymmetric Key - Use both ways - Notes

N O T E S

An asymmetric key can be used to both encrypt or decrypt data. This gives two slightly different uses as are depicted on this foil.

Anyone could send Bob a message, encrypting it with his public key. Doing this ensures that only Bob can decrypt the message with his private key - it is for his eyes only.

Bob could send a message, encrypting it with his private key. Everyone who gets hold of that message would be able to decrypt it using Bob's public key and be assured that the message had come from the owner of that private key - Bob.

Cryptography

- **Symmetric Keys - Secret Keys**
  - **Relatively fast**
  - **Poses key delivery challenges when faced with large numbers of senders/receivers**
  - **The key has to be known only by the sender/receiver...**
- **Asymmetric Keys - Public/Private Key Pairs**
  - **Message encrypted with one key can only be decrypted by the other one**
  - **Slower than secret-key cryptography**
  - **Designed to accommodate key delivery and scalability**

  - **Asymmetric Keys can be used to solve the key distribution challenges associated with symmetric keys**
- **Standard Key Sizes**

• **512 bits**	**Low-strength key**
• **768 bits**	**Medium-strength key**
• **1024 bits**	**High-strength key**

Cryptography - Notes

N O T E S

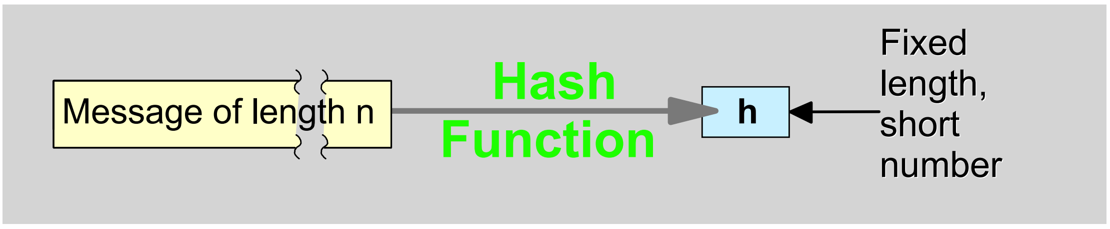
Both types of cryptography have benefits and costs. Symmetric keys can encrypt your data relatively fast in comparison to asymmetric keys. However, symmetric keys suffer from the key distribution problem. If you combine the two type of cryptography together, you can see that you could use assymmetric key cryptography to deliver your secret, therefore getting around the key distribution problem, and then use the faster symmetric key for your bulk data transmission.

We will see later that that is exactly what SSL does.

Although a key could be any size, the standard key sizes that are used are 512 bits which is a low-strength key, 768 bits and up to 1024 bits as a high-strength key.

Hash Function

- **Hash Function**
  - **Computes the message digest or Message Authentication Code (MAC)**
  - **Easy to compute**
  - **Very difficult to reverse**
  - **It should be computationally infeasible to find two messages that hash to the same thing.**



Hash Function - Notes

N O T E S

To be able to detect if someone has tampered with your message whilst in transit you can make use of a one-way hash function. The message is passed through this function which results in a short fixed length number which is unique to the contents of this message.

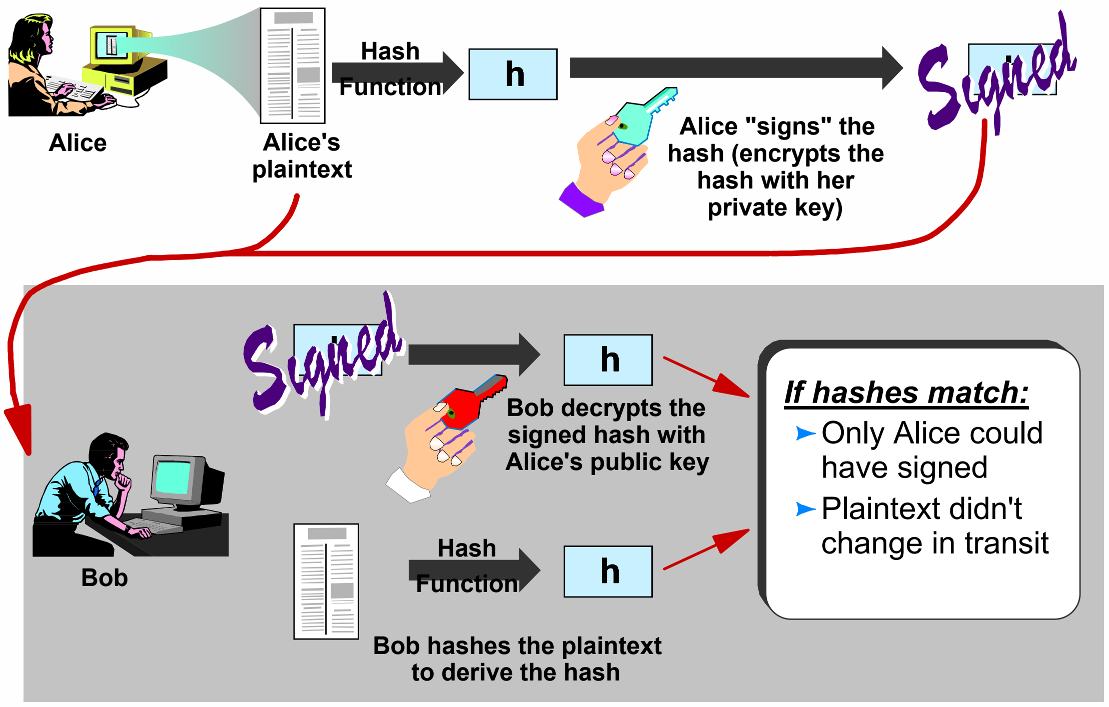
There is no way to determine the message from the one-way hash that is any simpler than going through all the possible values of the original message and computing the hash of each one.

The result is known as the message digest or message authentication code (MAC). When I send the message I also send the digest.

Any change to the message in transit will, with very high probability, result in a different message digest, therefore alerting the user to the change.

It is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.

Digital Signature



Alice
Alice's plaintext
Hash Function
h
Alice "signs" the hash (encrypts the hash with her private key)
Signed
Signed
h
Bob decrypts the signed hash with Alice's public key
Bob
Hash Function
h
Bob hashes the plaintext to derive the hash

If hashes match:

- Only Alice could have signed
- Plaintext didn't change in transit

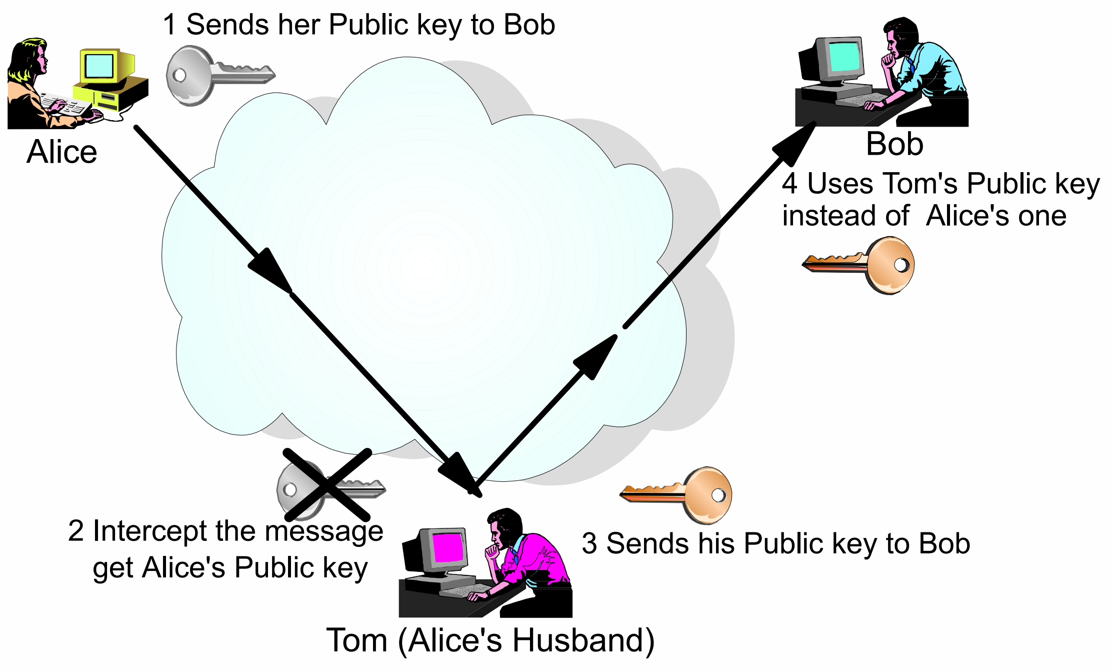
Digital Signatures - Notes

N O T E S

Digital signatures combine the use of the one-way hash function and public/private key encryption.

The message is hashed to provide a number, the hash number or message digest. This hash number is encrypted using Alice's private key to create the digital signature. The recipient of the message, in this case Bob, can also hash the message to get a hash number, and can decrypt the digital signature using Alice's public key, to get her hash number. If these numbers match then the message did come from Alice and also we know it hasn't been changed since it was signed.

Jealous Husband


1 Sends her Public key to Bob
Alice
Bob
4 Uses Tom's Public key instead of Alice's one
2 Intercept the message get Alice's Public key
3 Sends his Public key to Bob
Tom (Alice's Husband)

Jealous Husband - Notes

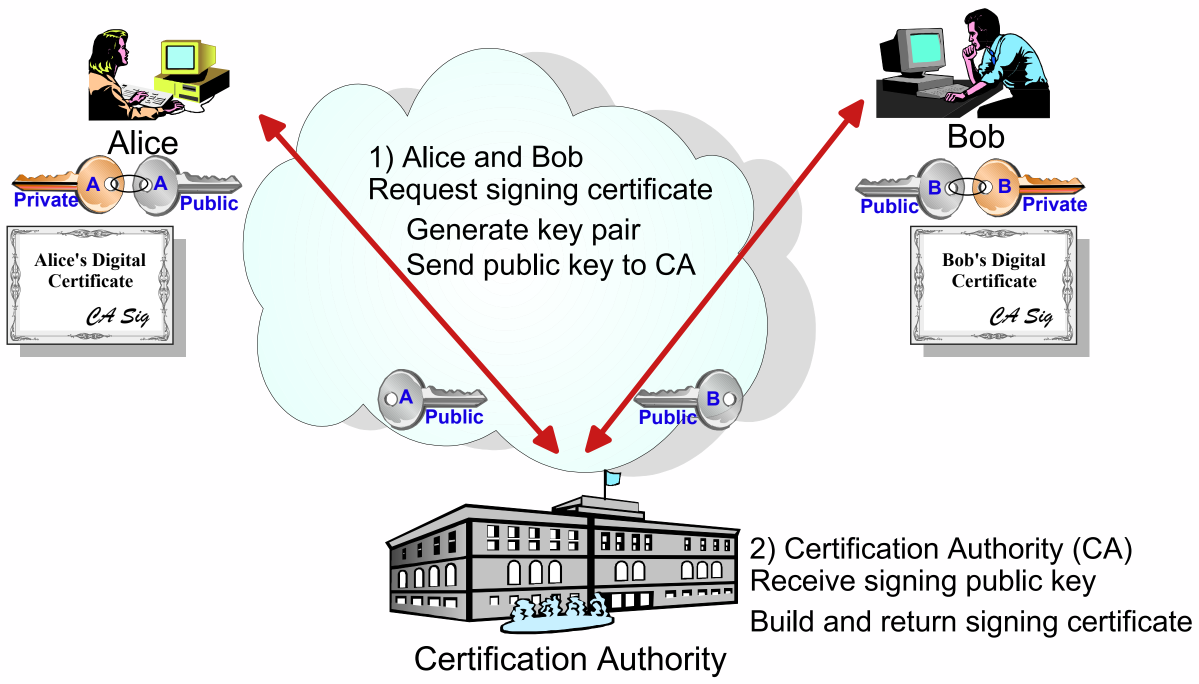
N O T E S

The initial transfer of public keys can be subject to a man-in-the-middle attack.

For example, Alice wants to send her public key to Bob so that he can use it to encrypt his transmissions to her. Alice's jealous husband, Tom, intercepts her message and replaces her public key with his own before sending it on to Bob. When Bob receives the key and encrypts his message with it, Tom can intercept the message and decrypt it with his private key to spy on them.

This then poses the question, how can I trust a public key?

Digital Certificate



Alice
Private
A
A
Public
Alice's Digital Certificate
CA Sig
1) Alice and Bob
Request signing certificate
Generate key pair
Send public key to CA
A
Public
B
Public
Bob
B
B
Public
Private
Bob's Digital Certificate
CA Sig
2) Certification Authority (CA)
Receive signing public key
Build and return signing certificate
Certification Authority

Digital Certificates - More info

- Certificate request
  - the sender's identity
    - Distinguished Name, well known format X.500 series
  - the sender's public key
  - generally money (though sometimes internal certification)
- Certificate, X.509 standard
  - the sender's verified identity
  - the sender's public key
  - the Certification Authority's digital signature
  - Expiry Date
- User Certificate
  - Create using Digital Certificate Management tool of choice
  - Binds an identity to a public key
- Certification Authority
  - Trustworthy Authority
  - "Well known" public key, to use for encryption of request for certificate

Distinguished Name

- **Well defined format**

CN="Morag Hughson" L=Hursley O=IBM OU="WebSphere MQ Development" C=England

CN	Common Name
T	Title
L	Locality name
ST/SP/S	State or Province name
O	Organisation name
OU	Organisational Unit name
C	Country

Digital Certificate - Notes

N O T E S

A Digital certificate contains information about the individual, for example their name and company, and also their public key. The certificate is signed, with a Digital Signature, by the Certification Authority (CA).which is a trustworthy authority.

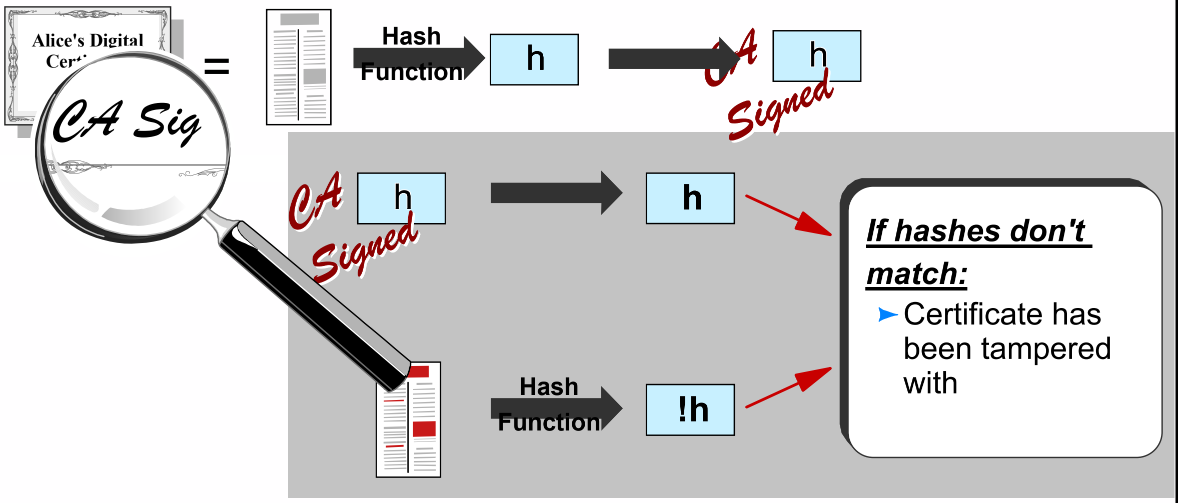
So in our example, Alice and Bob would need to request a signing certificate from the Certification Authority. Instead of sending each other their public keys, they would send them to the CA. The CA would verify the identity of the sender, and then create the certificates for Alice and Bob to use.

To obtain a digital certificate, one needs to send one's identity to the Certification Authority. This information is sent in a standard format that is defined by the X.500 series of standards. This identifying information is accompanied by the sender's public key. If the certificate is requested from an external CA such as Verisign, then the certificate will also cost money.

For testing purpose, or internal only use, it is also possible to have user certificates. These are not signed by an external CA, but instead, created and signed using whichever digital certificate management tool your company chooses to use, for example, RACF on z/OS. This has the major advantage that you don't have to pay an external CA for every certificate you produce. On the other hand, such certificates cannot be used to authenticate connections from outside the organization.

Trusting a Digital Certificate

- Digital Certificate = Plaintext
  - Can be subject to tampering
  - Signed by CA at creation



If hashes don't match:
- ➤ Certificate has been tampered with

- CA's Digital Signature
  - Allows tampering to be detected

Trusting a Digital Certificate - Notes

N O T E S

If Tom can intercept Alice's Key, can he not also intercept Alice's Certificate? How do I trust the certificate that has been sent to me? Surely someone could tamper with a certificate to pretend to be someone they're not?

This is where the CA signature comes into effect. A digital certificate can simply be thought of as a peice of plaintext that could be subject to tampering. After all it is just a file on your computer. How can we detect if someone has tampered with the certificate we are going to use.

Remember on the Digital Signature foil, we showed Alice signing her plaintext before sending it to Bob. Bob could then check the signature to ensure that Alice's message had not been tampered with. The same technique is used to determine whether a digital certificate has been tampered with.

The CA calculate the hash value of the plaintext (our certificate) and then signs that hash value with the CA private key to generate a CA digital signature. To check that the certificate is valid, the CA's digital signature can be decrypted using the CA public key (well known CA public keys are installed in many of the security products that use SSL) to check that the hash values match.

Secure Sockets Layer

- Protocol to allow transmission of secure data over an insecure network
- Combines these techniques
  - Symmetric / Secret Key encryption
  - Asymmetric / Public Key encryption
  - Digital Signature
  - Digital Certificates
- to combat security problems
  - Eavesdropping
    - Encryption techniques
  - Tampering
    - Digital Signature
  - Impersonation
    - Digital Certificates

Secure Sockets Layer - Notes

NOTES

Secure Sockets Layer (SSL) is an industry-standard protocol that provides a data security layer between application protocols and the communications layer, usually TCP/IP. The SSL protocol was designed by the Netscape Development Corporation, and is widely deployed in both Internet applications and intranet applications. SSL defines methods for data encryption, server authentication, message integrity, and client authentication for a TCP/IP connection. SSL uses public key and symmetric techniques to provide the following security services:

Message privacy
SSL uses a combination of public-key and symmetric key encryption to ensure message privacy. Before exchanging messages, an SSL server and SSL client perform an electronic handshake during which they agree to use a session key and an encryption algorithm. All messages between the client and the server are then encrypted. Encryption ensures that the message remains private even if eavesdroppers intercept it.

Message integrity
SSL uses the combination of a shared secret key and message hash functions. This ensures that nothing changes the content of a message as it travels between client and server.

Mutual authentication
During the initial SSL handshake, the server uses a public-key certificate to convince the client of the server’s identity. Optionally, the client may also exchange a public-key certificate with the server to ensure the authenticity of the client.

SSL Terms

Encryption
+
Hash Function
=
CipherSpec

CipherSpec
+
Authentication/Key Exchange
=
CipherSuite

SSL Terms - Notes

N O T E S

When we set up an SSL session, we can specify what encryption algorithm we wish to use. We can also specify what hash function to use to generate the message digest, or MAC (message authentication code). This combination is called the CipherSpec.

An SSL session also needs to know what algorithm to use for authentication and key exchange. In the current implementation of SSL that we will be using, the only option for this is RSA.

The combination of a authentication/key exchange algorithm and the CipherSpec is called a CipherSuite.

CipherSpecs

- Encryption
  - Block Cipher
    - RC2
    - DES
    - Triple DES
    - AES
  - Stream Cipher
    - RC4
- Hash Function
    - SHA
    - MD5

- CipherSpec
  - NULL_MD5
  - NULL_SHA
  - RC4_MD5_EXPORT
  - RC4_MD5_US
  - RC4_SHA_US
  - RC2_MD5_EXPORT
  - DES_SHA_EXPORT
  - RC4_56_SHA_EXPORT1024
  - DES_SHA_EXPORT1024
  - TRIPLE_DES_SHA_US
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

CipherSpecs - Notes

N O T E S

RC2, from RSA Data Security Inc. This is a block cipher algorithm (that is, it encrypts the data by blocks, 8-byte long) with a key length of 40 bits, 64 bits and 128 bits.

RC4, from RSA Data Security Inc. This is a stream cipher algorithm (that is, this algorithm operates on each byte of data, as opposed to a block of bytes) with a key length of 40 bits, 64 bits and 128 bits.

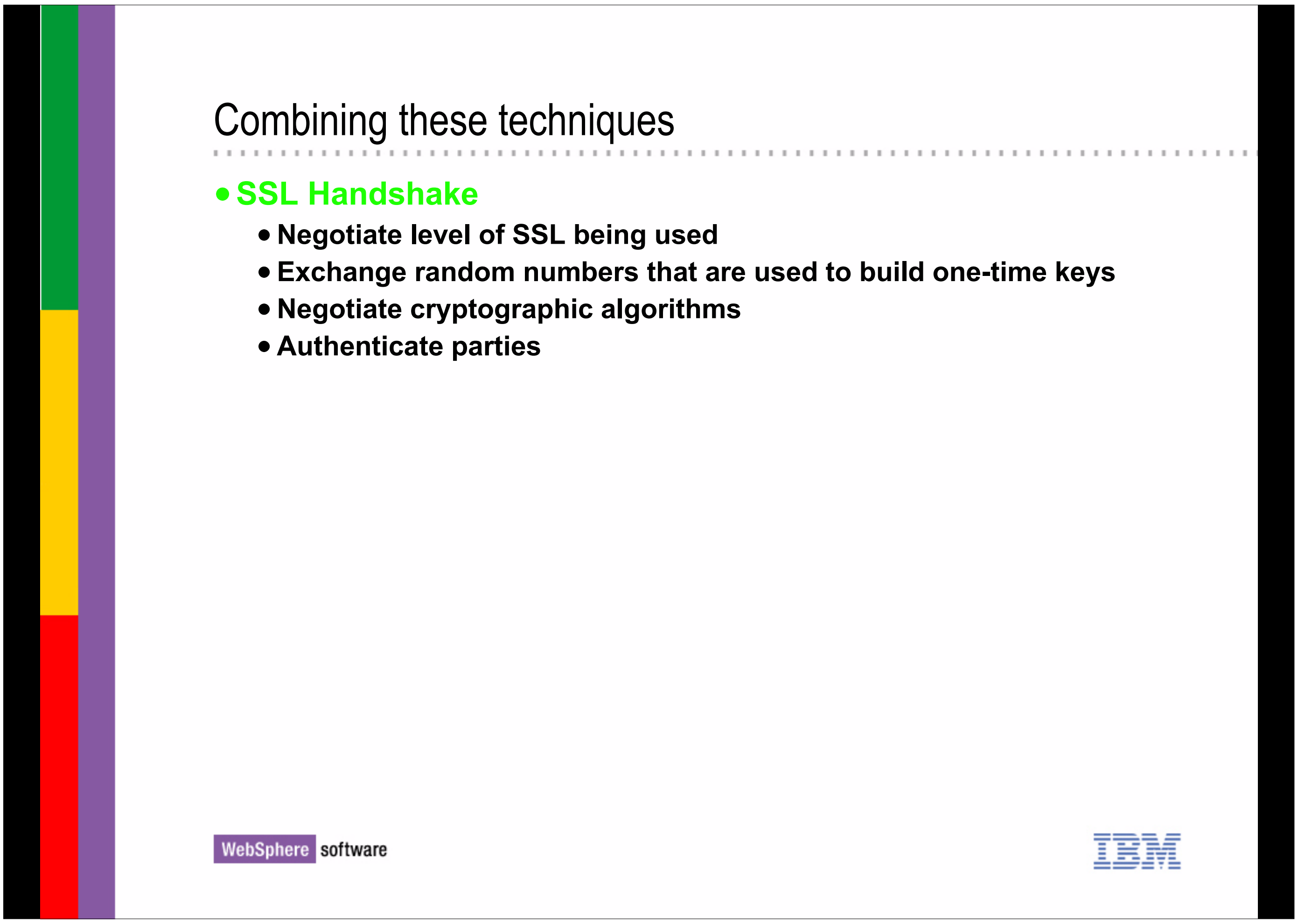
DES, the old US Data Encryption Standard. This is a block cipher algorithm, with 8-byte long blocks and a key length of 56 bits.

Triple DES is a variation of DES, with a key length of 168 bits

In order fastest to slowest they are: RC4, DES, RC2, TripleDES.

AES, Advanced Encryption Standard, the new US standard. This is a block cipher algorithm, with 16-byte long blocks. Key lengths include 128 and 256 bits.

There are two choices of hash function that can be used in the provided selection of CipherSpecs. These are SHA and MD5. SHA stands for Secure Hash Algorithm and MD5 stands for Message Digest (version) 5. The choice between MD5 and SHA-1 is a trade off between security and performance. The SHA algorithm produces a 160-bit output and the MD5 algorithm only a 128-bit output. This is compared with the fact that the MD5 algorithm is much faster in calculating the message digest.



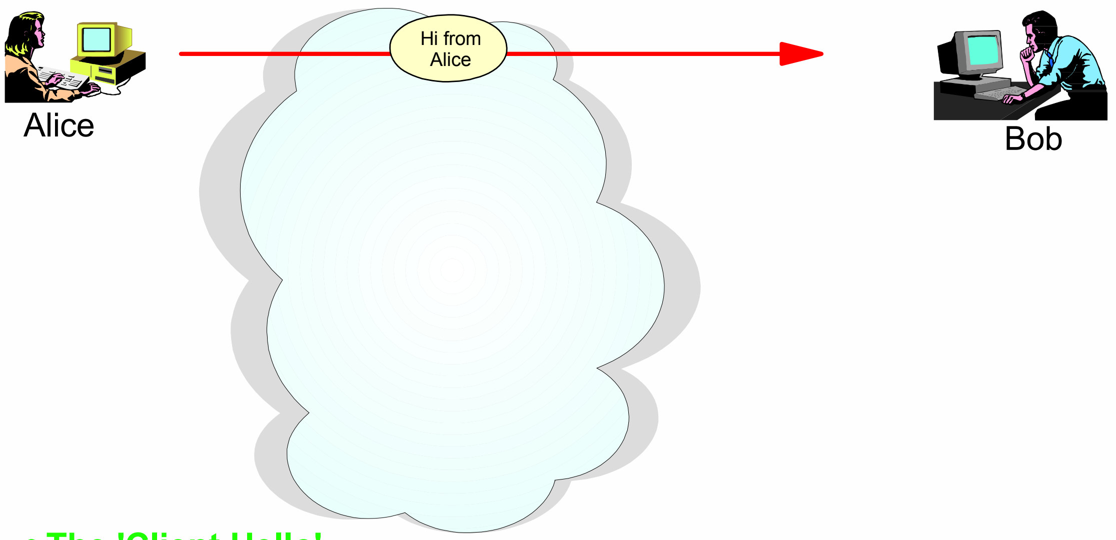
Combining these techniques

- SSL Handshake
  - Negotiate level of SSL being used
  - Exchange random numbers that are used to build one-time keys
  - Negotiate cryptographic algorithms
  - Authenticate parties

WebSphere software

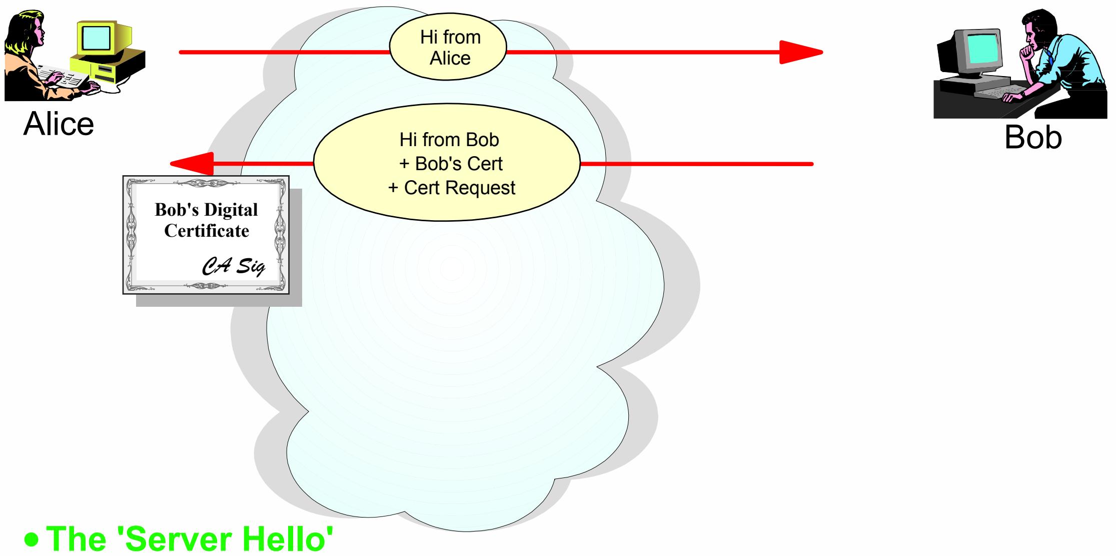
IBM

SSL Handshake



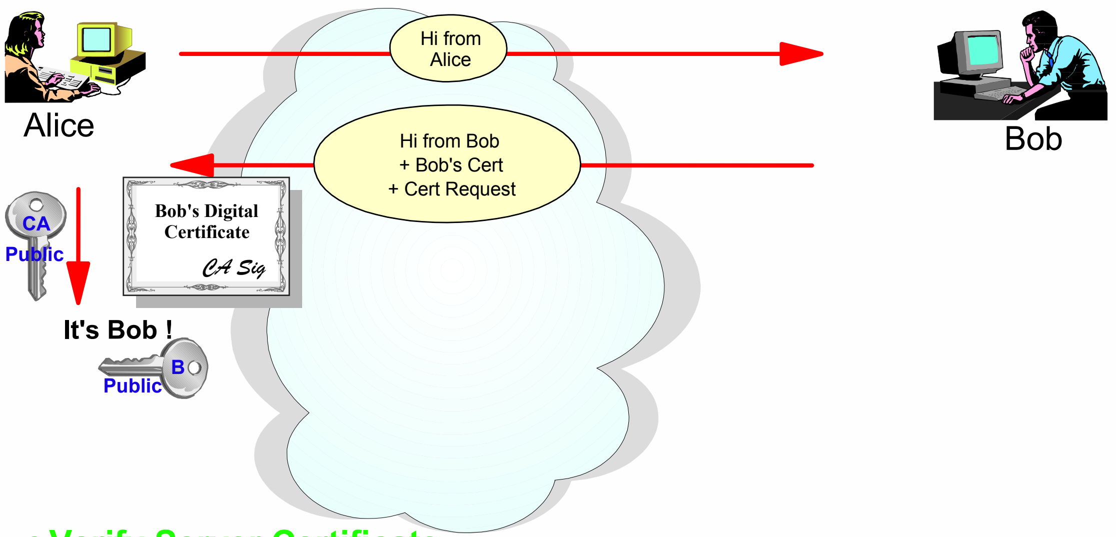
- **The 'Client Hello'**
  - **Alice sends Bob some random text**
  - **Also sends what CipherSpecs and compression methods she can use**
  - **Alice is considered the client since she started the handshake**

SSL Handshake



- **The 'Server Hello'**
  - **Bob sends Alice some random text**
  - **Bob chooses the CipherSpec be used, from Alice's list**
- **The Server Certificate**
- **The Client Certificate Request**

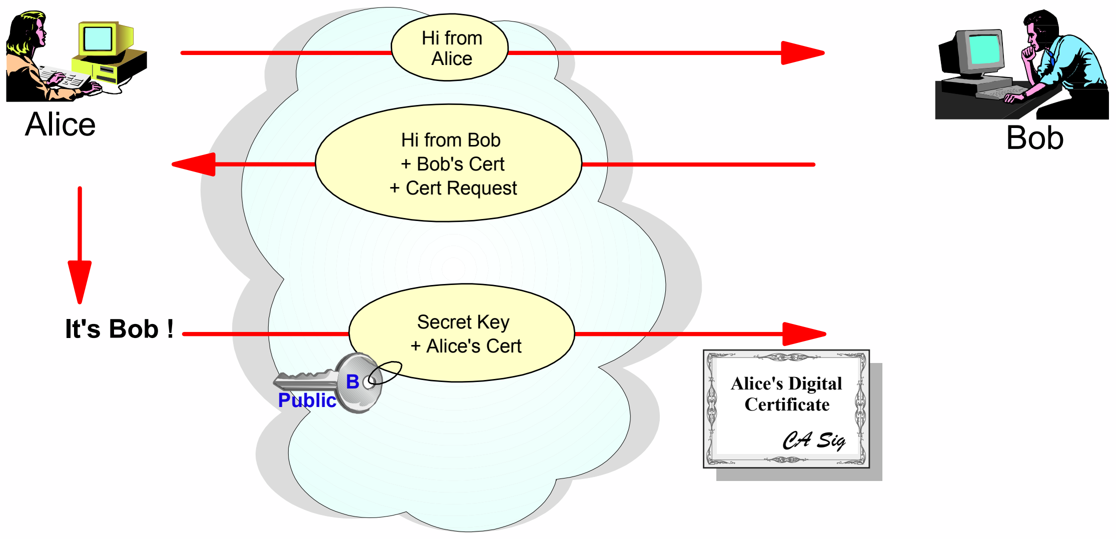
SSL Handshake



- **Verify Server Certificate**
  - **Check Validity Period**
  - **Decrypt using CA's Public Key - verifies that CA is trusted**
  - **Check Domain Name and/or Distinguished Name**
  - **Also receives Bob's Public Key**

WebSphere software

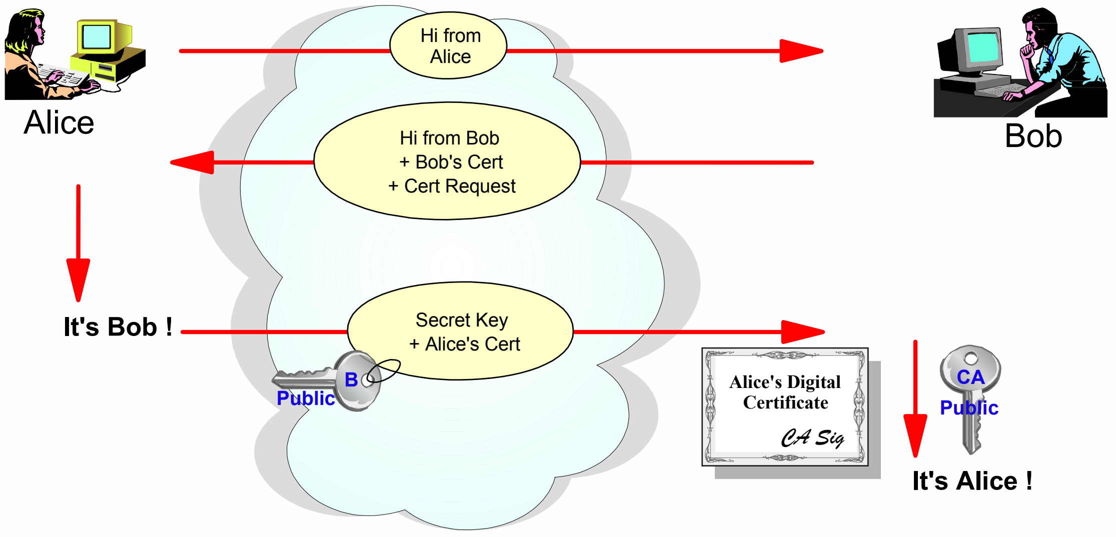
SSL Handshake



- **Client Key Exchange**
  - **Alice sends Bob the Secret Key to use**
  - **This is encrypted with Bob's Public Key**
  - **Also sends her certificate**

WebSphere software

SSL Handshake

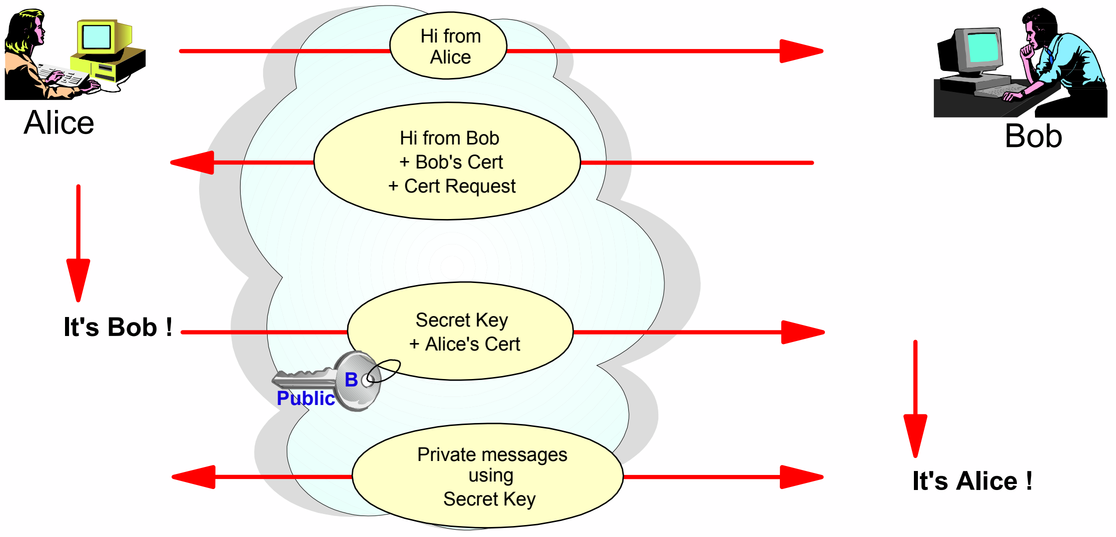


- **Verify Client Certificate**
  - **Decrypt using CA's public Key**

WebSphere software

IBM

SSL Handshake



- **Send Information using agreed Secret Key**
  - **Randomly generated 1-time key**
- **This is now a 'secure line'**

WebSphere software

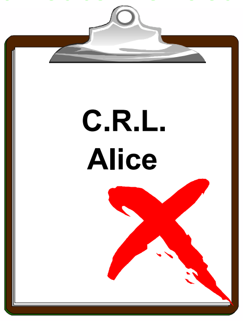
IBM

Certificate Revocation

- **What happens if a Certificate is no longer trusted?**

Valid From	01/04/2002
Valid To	01/10/2002

- **Certification Authority revokes it on a Certificate Revocation List (CRL)**



Certificate Revocation - Notes

N O T E S

A digital certificate has two dates associated with it. It has a date from which it is valid and it has a date after which it is invalid, an expiry date. What happens in the circumstance where a certificate has been issued and before its expiry date is reached, its status is considered to be no longer trusted?

A Certification Authority can revoke a certificate which is no longer trusted by publishing it in a Certificate Revocation List (CRL). When a certificate is received it can be checked against this list to ensure that it has not been revoked.

Benefits of SSL

- Provides a protocol for the function we need
  - Encryption
  - Message Integrity Checking
  - Authentication
- Supports a range of cryptographic algorithms
- Uses Public/Private Keys
  - No key distribution problem
- Widely accepted in the Internet community
- Subjected to significant testing by the hacker community

Benefits of SSL - Notes

NOTES

The Secure Sockets Layer (SSL) provides authentication, message integrity checking and data encryption for messages sent across the Internet. It has become the de facto standard for Internet security and is widely available on different operating systems.

As we have already seen, it supports a wide range of cryptographic algorithms and makes use of public/private keys for authentication which removes the need for an online key distribution center.

SSL functions and WebSphere MQ

- **Supported**
  - **SSL V3.0**
  - **Choice to authenticate client**
  - **Certificate Revocation Lists on LDAP servers**
- **Not Supported**
  - **List of CipherSpecs, only one must be provided**
  - **SSL session reuse**

SSL functions and WebSphere MQ - Notes

N O T E S

WebSphere MQ will be using SSL V3.0. SSL V3.0 was introduced in 1996; lower levels of SSL are no longer widely used.

On the SSL handshake an optional feature is to authenticate the client certificate as well as the server certificate. WebSphere MQ offers this choice.

The CRL function of SSL will be supported in WebSphere MQ via LDAP servers.

SSL can allow lots of CipherSpecs to be passed by the client on the SSL handshake and the server will choose one that it supports. WebSphere MQ will impose the restriction that only one CipherSpec can be supplied on the channel definition which must match at both ends.

SSL can allow its sessions to be reused. This could be useful for short lived web queries. However, since WebSphere MQ channels are likely to be longer running, we will not be using this feature of SSL.

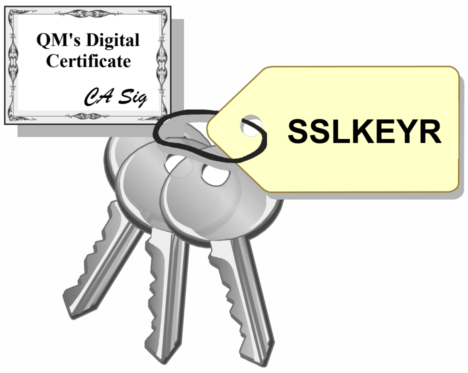
WebSphere MQ Configuration

WebSphere MQ Configuration Tasks

- Associating a certificate with a queue manager
- Associating a certificate with an WebSphere MQ client
- Allowing access to Certificate Revocation Lists (CRLs)
- Specifying cryptographic hardware (some platforms)
- Specifying SSL tasks (z/OS)
- New channel attributes
  - Specifying CipherSpec
  - Specifying permitted partners
  - Specifying that the partner must provide a certificate

Queue Manager's Key Repository

- Queue Manager's own Digital Certificate
  - ibmWebSphereMQ<QMgr Name> (mixed case) label on z/OS
  - ibmwebspheremq<qmgr name> (lower case) label on UNIX and OS/400
  - Selected from a GUI on Windows
- Digital Certificates from various Certification Authorities



- On z/OS

```
ALTER QMGR SSLKEYR(CSQ1RING)
```

- On Unix, Windows, OS/400

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

Queue Manager's Key Repository - 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 a RACF (z/OS) or iKeyMan (UNIX) .

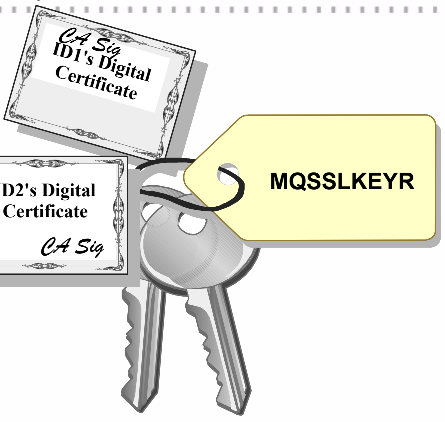
On z/OS, the required certificate in the key repository is specified with the mixed-case label ibmWebSphereMQ<QMgr Name>. On UNIX and OS/400, the required certificate in the key repository is specified with the lower-case label ibmwebspheremq<qmgr name> . Note that the certificate label is also sometimes referred to as its "friendly name". Selection on Windows is achieved using a GUI to assign the certificate to the queue manager.

The key repository generally also contains a number of signed digital certificates from various Certification Authorities which allows it to verify certificates it receives from its partner at the remote end of the connection.

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 or certificate store.

WebSphere MQ Client's Key Repository

- Client's own Digital Certificate
  - ibmwebspheremq<logon userid> (lower case) label on UNIX clients
  - Selected from a list on Windows
- Digital Certificates from various Certification Authorities



- Specify
  - Environment variable:

export MQSSLKEYR=var/mqm/ssl/key

  - MQCONNX

SSLKeyRepository

WebSphere MQ Client's Certificate

N O T E S

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.

A particular personal certificate within that file is selected for use on the client's SSL channels.

- UNIX clients use the certificate labeled with ibmwebspheremq followed by the logon userid, wrapped to lower case.

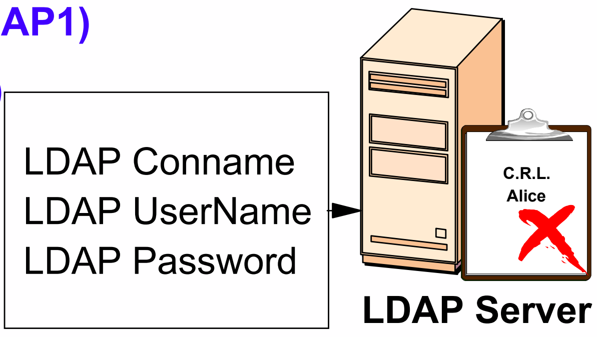
- Windows clients use a certificate selected from a list using a numeric handle.

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.

Queue Manager: Access to Certificate Revocation Lists

- Define AUTHINFO objects

```
DEFINE AUTHINFO(LDAP1)
AUTHTYPE(CRLLDAP)
CONNAME(...)
LDAPUSER(...)
LDAPPWD(...)
```



- Put these AUTHINFO objects into a namelist

```
DEFINE NL(LDAPNL) NAMES(LDAP1, LDAP2, ...)
```

- Associate namelist with QMGR

```
ALTER QMGR SSLCRLNL(LDAPNL)
```

Queue Manager: Allow Access to CRLs - Notes

N O T E S

Certificate Revocation Lists (CRLs) 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.

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 and the password associated with the Distinguished Name. The Distinguished name used here is in the same format as has been discussed earlier in this presentation. Note that LDAP CRL servers are generally defined to be publicly readable.

These parameters are defined on a new type of 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. On the distributed platforms up to ten of these AUTHINFO objects can be supplied for CRL checking. On z/OS there can currently be only one.

The list of AUTHINFO objects are named in a namelist and this namelist is specified on the SSLCRLNL qmgr 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); details are provided in the "SSL for iSeries" presentation.

WebSphere MQ client: Certificate Revocation Lists

- Record in the client channel definition table
  - Definitions which were current on the queue manager when the table was copied off
- Also on MQCONNX
- Also, on Windows, can specify in the Active Directory

WebSphere MQ client: Allow Access to CRLs - Notes

N O T E S

When the channel runs the client channel definition table CRL information does not have to match the definitions current on the queue manager system at that stage.

MQCONNX provides a new structure, MQAIR (MQ auth info records), to allow LDAP CRL information to be specified.

LDAP CRL records are created in the Active Directory using setmqcrl, a command-line queue manager command which is only supported on Windows. Websphere MQ clients which have access to the Active Directory read these records.

Active Directory support is only integral on Windows 2000 and above. Windows NT can use the Active Directory if the Microsoft Active Directory client extensions are applied.

CryptoGraphic Hardware on the UNIX platforms

- Parameters
  - are required by the SSL support.
  - required vary according to hardware used
  - only apply to UNIX platforms
- Specify

  ALTER QMGR SSLCRYP(<string>)
  - <string>: hardware involved + SSL support parameters
- On MQ client:
  - Environment variable

  SET MQSSLCRYP=<string>
  - MQCONNX

  SSLCryptoHardware

CryptoGraphic Hardware on the UNIX platforms: Notes

N O T E S

SSLCryptoHardware is in the new 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 Tasks

- MVS Tasks
  - To run SSL handshake and encryption calls
  - At least 2 required to run any SSL channels
- On z/OS

  ALTER QMGR SSLTASKS(8)

SSL Tasks - Notes

N O T E S

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 zero of these tasks then no SSL channels will be able to start.

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 of the channel must specify the same CipherSpec
- From a list of human-readable strings
  - e.g. NULL_MD5
  - RC4_MD5_US
- z/OS, Windows, OS/400: also SSL API numeric values
  - Allow support of new CipherSpecs without updates to MQ Code

```
SSLCIPH(RC4_MD5_US)
or
SSLCIPH(04)
```

SSLPEER

- Specify the partner's Distinguished Name
- Can use wildcards
- Multiple Organisational Unit (OU)
  - Must be matched in order

```
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 new 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 shown earlier in the presentation 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, Windows and OS/400, 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 as the UNIX SSL support is part of Websphere MQ.

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.

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.

Security Administration Tasks

- Creating certificates and Certificate Requests
  - Contains a Distinguished Name
  - Different tools on various platforms
- Storing Certificates & Public Keys and Private Keys
  - Keyrings in RACF, ACF2 or TopSecret on z/OS
  - Key database files on UNIX platforms and OS/400
  - Certificate Stores on Windows
    - private keys are stored in the Registry on Windows
- Managing certificates cross platform
  - Binary, must be transfered in binary
    - DER, CER, BER encodings, e.g.PKCS #7 DER encoded X.509 certificate
    - PKCS #12 DER encoded X.509 certificate (password protected). PKCS #12 files contain a personal certificate and, optionally, its private key and CA certificate(s) for its signing CA(s)
  - Text
    - must be transfered with text conversion, e.g. ACSII -> EBCDIC
    - Privacy Enhanced Mail (PEM) encoded X.509 certificate
    - Base64 encoded certificate

Security Administration Tasks - Notes

N O T E S

Once created, digital certificates are stored in a key repository. These key repositories are different on the different plaforms. They are:

in the External Security Manager's (ESM's) database on z/OS, e.g. in RACF, and are connected to key rings

in a key database file on the UNIX platforms and OS/400

in a certificate store on Windows platforms

Certification Authority certificates are also stored in your key repository so they can be used to validate certificates received from the partner system at the remote end of the connection.

Certificates can be created on the system, or they can come from outside the system and need to be imported onto the system. There are several standard formats that these can be in. Some of these formats are binary formats and must be transported in their exact binary format. In contrast, text formats must be transported as text and if transported between an ASCII and EBCDIC system, the ASCII to EBCDIC translation must be performed.

Creating Certificates

- **Certificates Contain the Distinguished Name**

  CN="Morag Hughson" L=Hursley O=IBM OU="WebSphere MQ "
  C=England

- **Create internal test certificates,**
  - **Using RACF panels or RACDCERT commands on z/OS**
  - **Using iKeyMan GUI tool on Unix platforms**
  - **Using Microsoft MAKECERT tool on Windows**
  - **Using Digital Certificate Manager (DCM) on OS/400**

OR

- **Generate a certificate request**
  - **This request is written to a file/data set**
  - **Send it to the Certification Authority; this may be via their website**
  - **Receive your signed certificate from the CA**
- **Import Certificate into repository**
  - **labelled appropriately (z/OS, UNIX and OS/400)**
    - **'ibmWebSphereMQ<qmgr-name>' (z/OS)**
    - **'ibmwebspheremq<lower-case-qmgr-name>' (UNIX and OS/400)**

Creating Certificates - Notes

N O T E S

A digital certificate can be created on your own system and self-signed or signed by your site's Certificate Authority (this may be useful for internal use certificate or for testing purposes).

If you wish to communicate with an external entity, however, you may need to get a certificate signed by a Certification Authority. To do this you generate a certificate request, or make a request based on an existing certificate in your repository, and send it to the CA. Once you receive a signed certificate you import it back into your repository and proceed to use it.

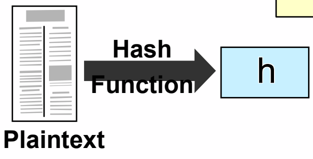
The certificates are associated with individual queue managers using a label based on the name of the queue manager.

Security ~~Problems~~ Solutions Using WebSphere MQ

- **Eavesdropping**
  - **Symmetric Key Cryptography**

SSLCIPH(RC4_MD5_US)

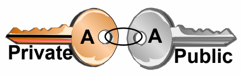
- **Tampering**
  - **Hash Function**



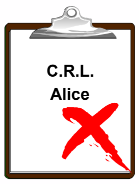
- **Impersonation**
  - **Digital Certificates**

SSLKEYR(QM1KEYRING)
SSLPEER('O=IBM')
SSLCAUTH(REQUIRED)

  - **Asymmetric Keys**



  - **CRL checking**



SSLCRLNL(LDAPNL)

Security Solutions with WebSphere MQ - Notes

N O T E S

In this presentation we have talked about three main security problems, eavesdropping, tampering and impersonation.

We have shown 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 lists.

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 which can be specified to WebSphere MQ. We can also check that we are talking to the partner we expect to be talking to 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.