Digital Certificates & TLS Handshake Educational Package

John J. Brefach
Washington Systems Center
z/OS Connect SME: Mitch Johnson
mitchj@us.ibm.com

Washington Systems Center

Agenda


Introduction to Encryption


Transport Layer Security


Anatomy of a Digital Certificate

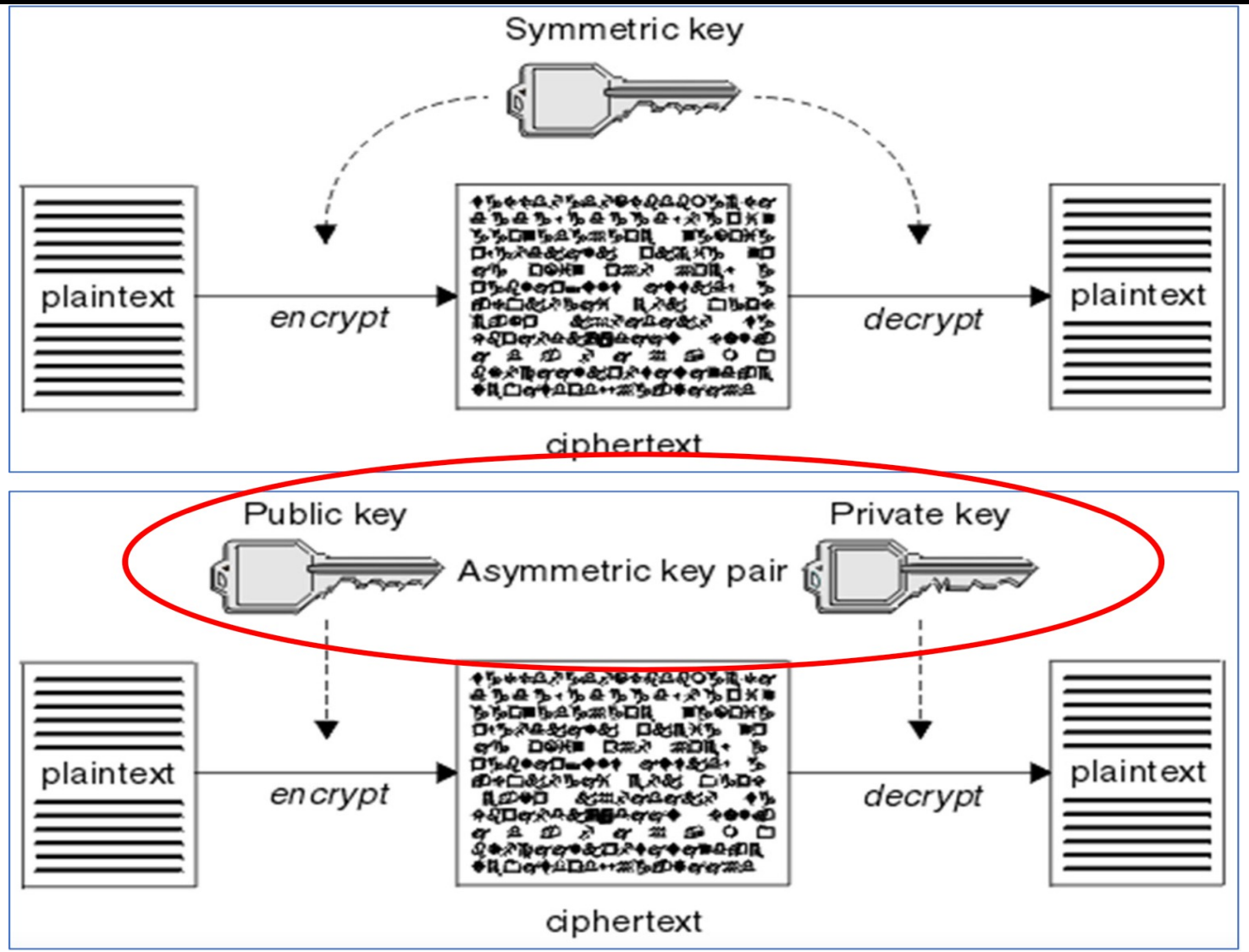

TLS Handshake Protocol


Problem Tracing options


Q&A

John.J.Brefach@ibm.com

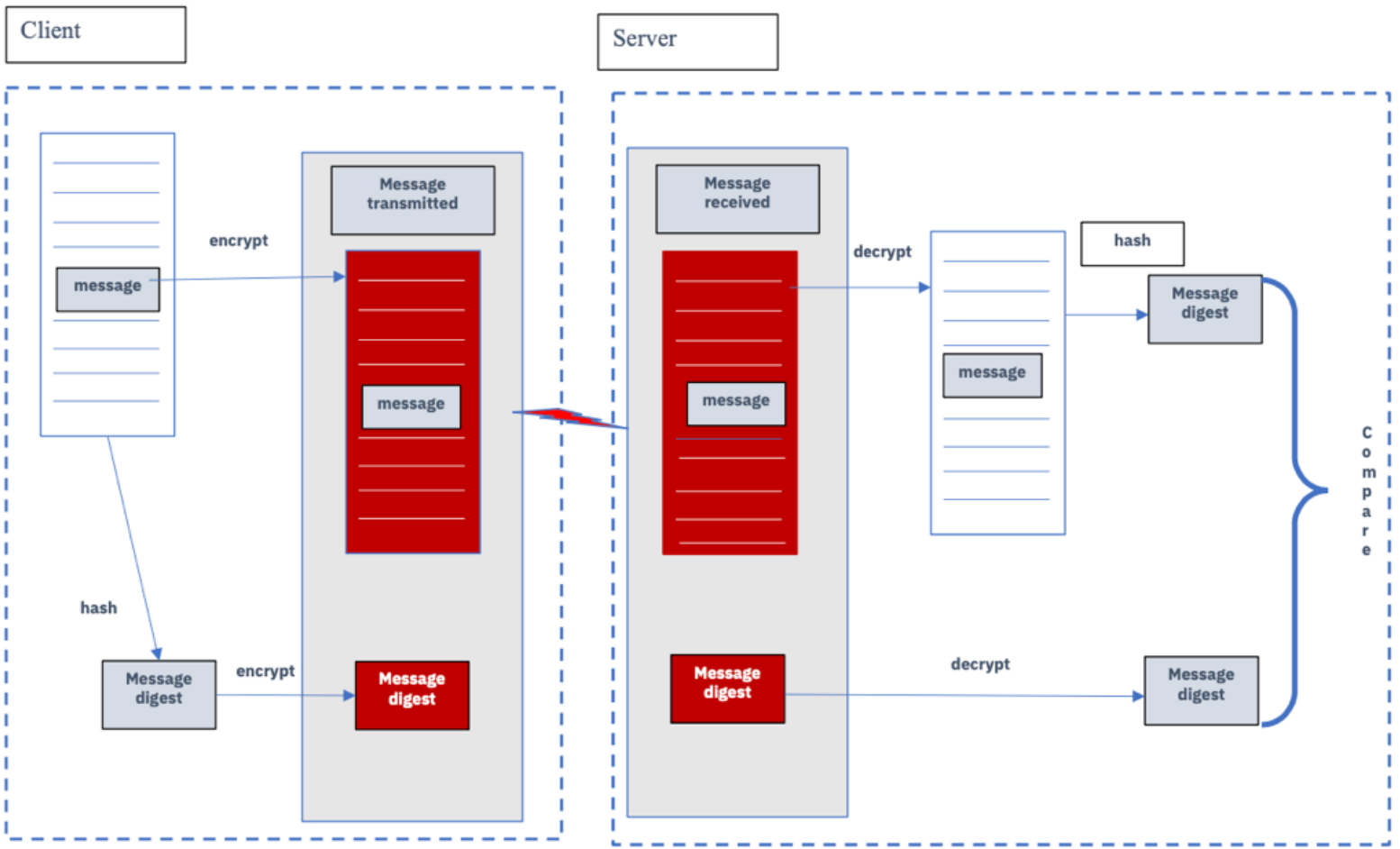
Introduction to Encryption



John.J.Brefach@ibm.com

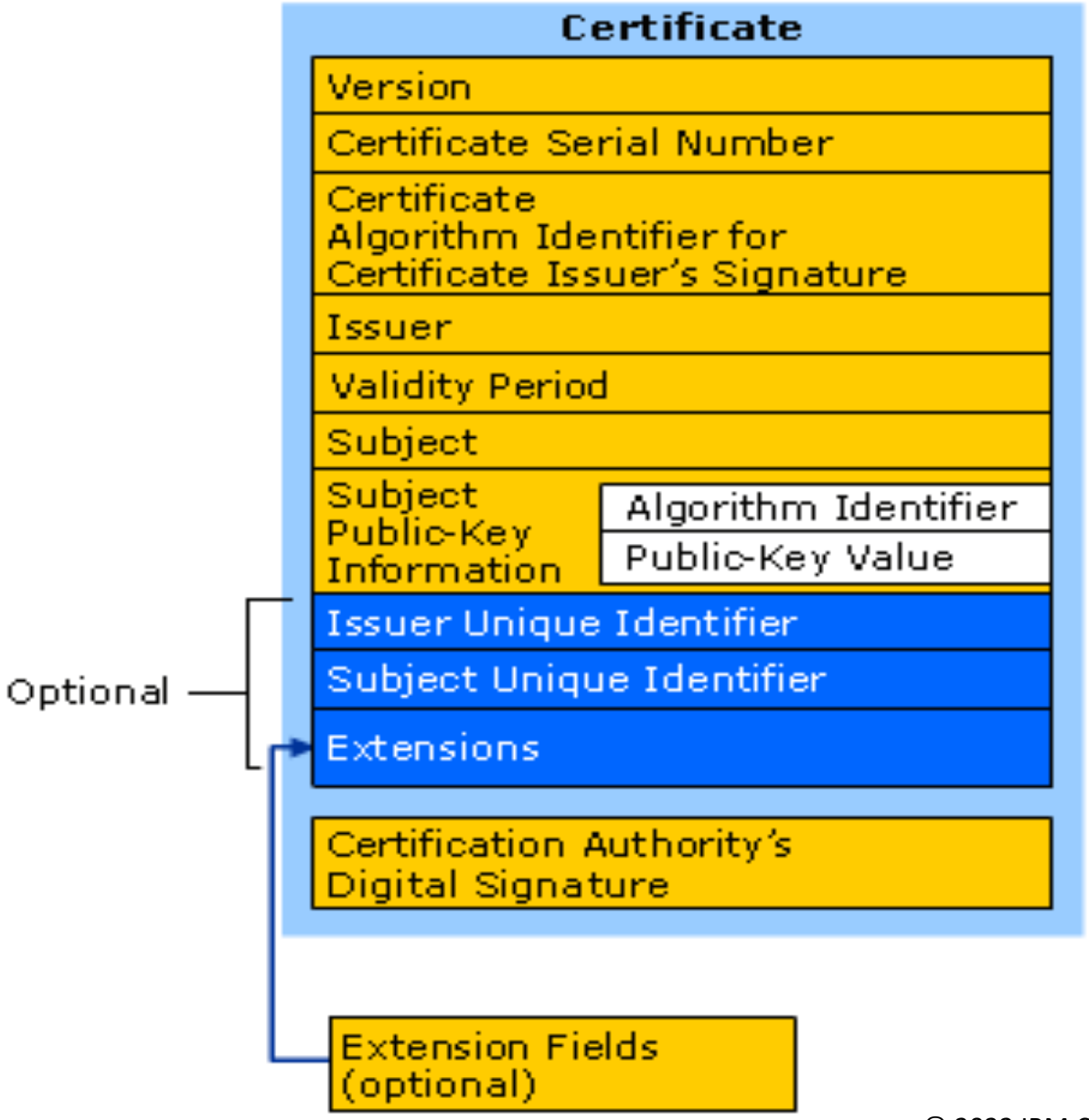
Transport Layer Security (TLS)

- Agreed upon protocol to exchange information
- TLS protocol utilizes Digital Certificates



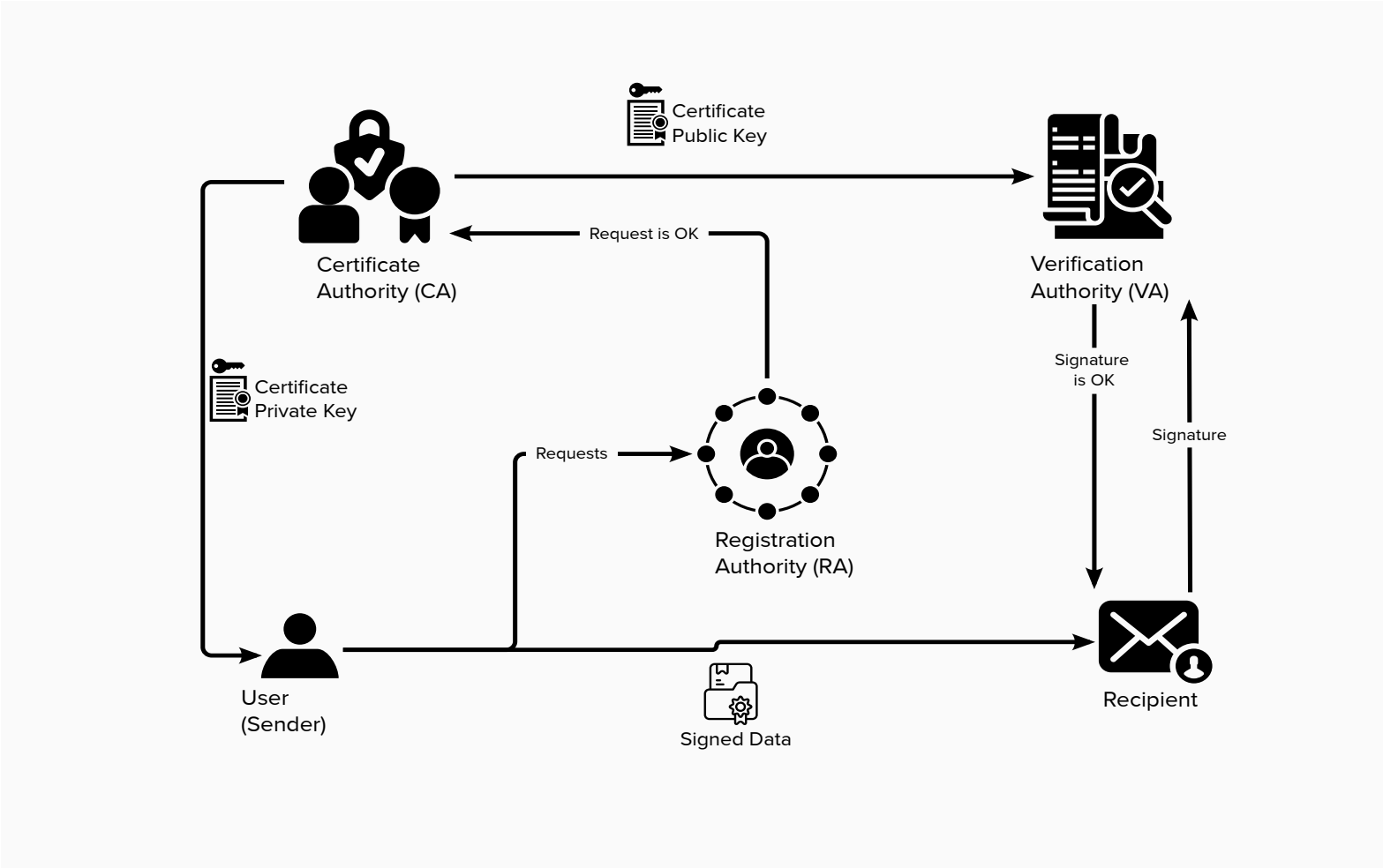
John.J.Brefach@ibm.com

Anatomy of Digital Certificate



John.J.Brefach@ibm.com

Components of Public Key Infrastructure



John.J.Brefach@ibm.com

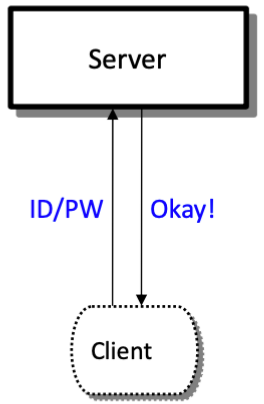
z/OS Connect Security Options

- Key Security Considerations
  - Authentication
  - Data integrity
  - Encryption
  - Authorization

Liberty Authentication Options

Several different ways this can be accomplished:

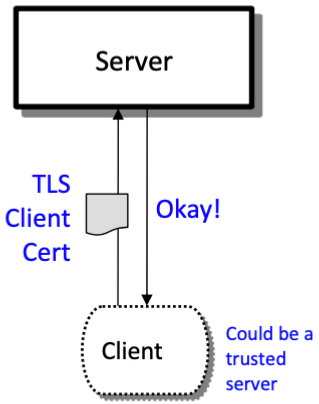
Basic Authentication



Client supplies ID/PW or ID/PassTicket

Server checks registry:
- **Basic (server.xml)**
- **SAF**

Client Certificate

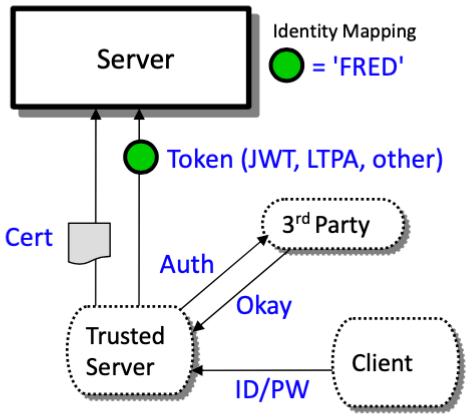


Client supplies client personal certificate

Server validates client personal certificate and maps it to an identity

Registry options:
- **SAF**

Third Party Authentication



Client authenticates to 3rd party sever

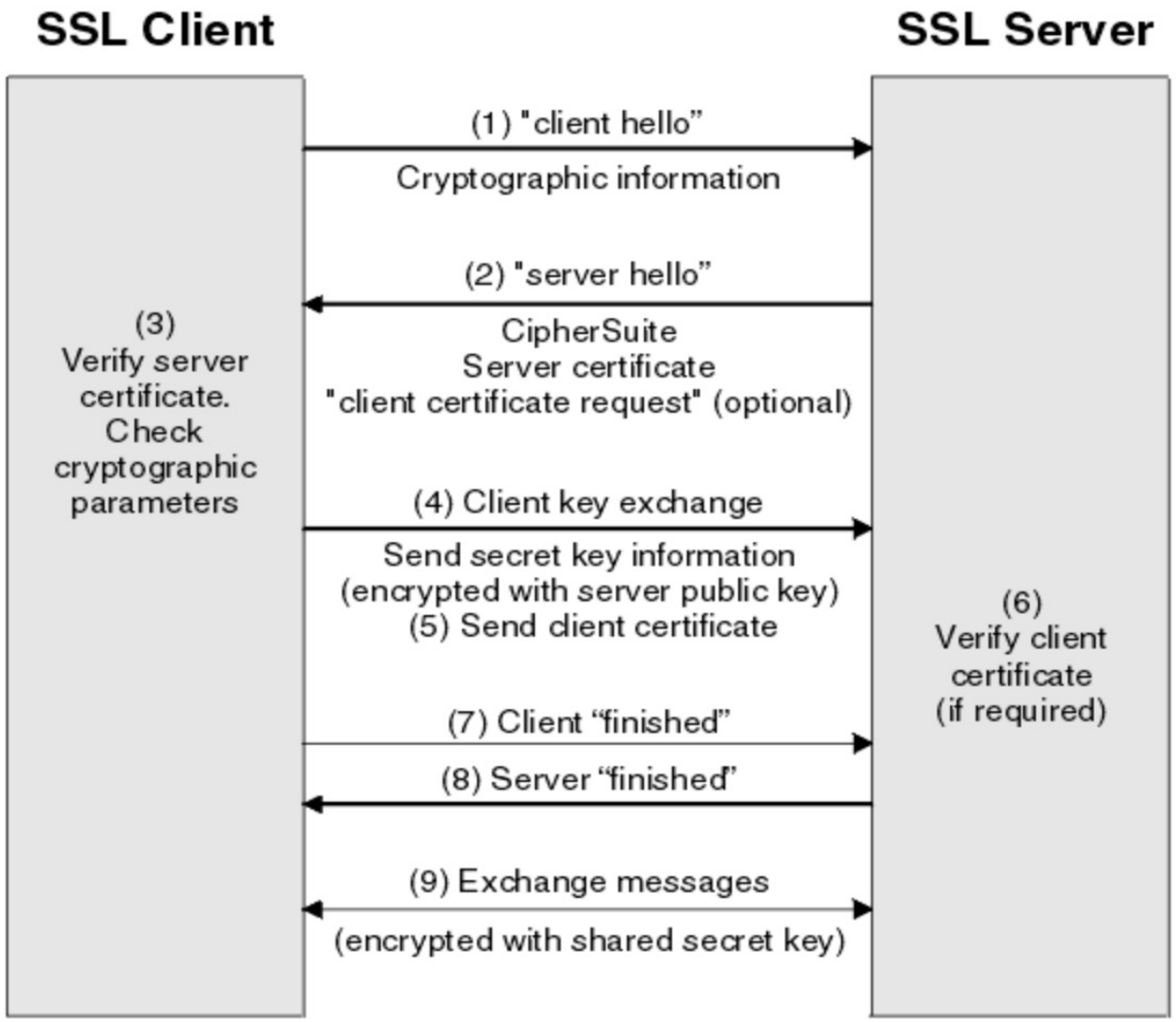
Client receives a trusted 3rd party token

Token flows to server and is mapped to an identity

Registry options:
- **We may not need to know these details.**

John.J.Brefach@ibm.com

How it flows (TLS Handshake)



John.J.Brefach@ibm.com

Low-level Overview

1. A Client sends a request to server for a protected session in a ***ClientHello*** message. Included in the request is the TLS capabilities of the client (e.g., TLS 1.2 or 1.3) and a list of supported ciphers in preference order.
2. The server selects the TLS version and selects cipher from the list sent by the client and returns this information in a ***ServerHello*** message.
3. The server's certificate (including the **public key**) is sent to the client in a ***Certificate*** message.
4. The server sends cryptographic information for the client to use for encrypting a pre-master key in a **Server key exchange** message.
5. **For mutual authentication, the server sends a *CertificateRequest* message requesting a client's personal certificate.**
6. The server concludes by sending a ***ServerHelloDone*** message.
7. The client verifies the server's certificate with its trust store.
8. **If mutual authentication is requested, the client sends its personal certificate in a *Certificate* message**
9. The client then uses the ***server's public* key** to generate and encrypt a 48 byte "premaster secret" message which is sent to the server in a ***ClientKeyExchange*** message.
10. **When mutual authentication is requested, a digitally signature (hashed) of the concatenation of all previous handshake messages is encrypted with the client's private key sent in a *CertificateVerify* message.**
11. The ***Change Cipher*** message is used to change the from cipher used during the handshake so all subsequent messages will be encrypted using a different cipher.
12. The server uses its **private key** to decrypt the "premaster secret" message (**only the private key can be used to decrypt the message**).
13. **If mutual authentication is requested, the server verifies the client's personal certificate with its key ring and uses the client's public key to decrypt and verify the message sent in the *CertificateVerify* message.**
14. Both the Client and Server use the "premaster secret" to compute a 'master secret', also know as "shared secret" or "session key" (symmetric encryption)
15. Client and server will use this "shared secret" or "session key" to encrypts messages sent between the endpoints.

John.J.Brefach@ibm.com

Problem Tracing Options

- Difficult to trace a TLS handshake as there is no log of messages being sent without using a trace
- Trace options will vary from one subsystem to another, they are all using different code to use the TLS protocol
  - Java, AT-TLS, Liberty, Global Security Toolkit
- Knowing the anatomy of Digital certificates is essential
- Knowing the TLS Handshake flow

Managing trace specifications

- Use “include” file to save commonly used trace specifications.
- Add the “include” after the sever has started to avoid tracing the startup activity.

server.xml

```
<include location="${server.config.dir}/includes/safTrace.xml"/>
```

safTrace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="security trace">
<logging traceSpecification="com.ibm.ws.security.*=all:
         SSLChannel=all:SSL=all:zosConnectSaf=all:zosConnect=all"/>
</server>
```

cicsTrace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="CICS trace">
<logging traceSpecification="zosConnectServiceCics=all:
         com.ibm.zosconnect.wv*=FINEST:zosConnect=all"/>
</server>
```

imsTrace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="IMS trace">
<logging traceSpecification="com.ibm.ims.*=all:
         com.ibm.j2ca.RAIMSTM=all:com.ibm.zosconnect.wv*=FINEST:
         zosConnect=all"/>
</server>
```

Enables enhanced tracing

(after adding an “include” file)

```
F BAQSTRT,REFRESH,CONFIG
```

Disable enhanced tracing

```
F BAQSTRT,LOGGING='*=INFO’
```

Or

```
F BAQSTRT,REFRESH,CONFIG
```

(after removing the “include” file

https://www.ibm.com/support/pages/collecting-troubleshooting-data-zos-connect-enterprise-edition

John.J.Brefach@ibm.com

cURL trace to show the flow with mutual authentication

```
* successfully set certificate verify locations:
* TLSv1.3 (OUT), TLS handshake, Client hello (01):
* TLSv1.3 (IN), TLS handshake, Server hello (02):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, CERT verify (15):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (01):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
*  subject: O=IBM; OU=LIBERTY; CN=wg31.washington.ibm.com
*  start date: Jan  4 04:00:00 2021 GMT
*  expire date: Jan  1 03:59:59 2023 GMT
*  common name: wg31.washington.ibm.com (matched)
*  issuer: OU=LIBERTY; CN=CA for Liberty
*  SSL certificate verify ok.
```

```
enum {
hello_request(0),
client_hello(1),
server_hello(2),
certificate(11),
server_key_exchange (12),
certificate_request(13),
server_hello_done(14),
certificate_verify(15),
client_key_exchange(16),
finished(20),
 (255) }
 HandshakeType;
```

* TLS 1.2 https://tools.ietf.org/html/rfc5246
TLS 1.3 https://tools.ietf.org/html/rfc8446

John.J.Brefach@ibm.com

Use the Java directive javax.net.debug to enable Java SSL tracing

Add this directive to the JVM properties *-Djavax.net.debug=ssl,handshake*

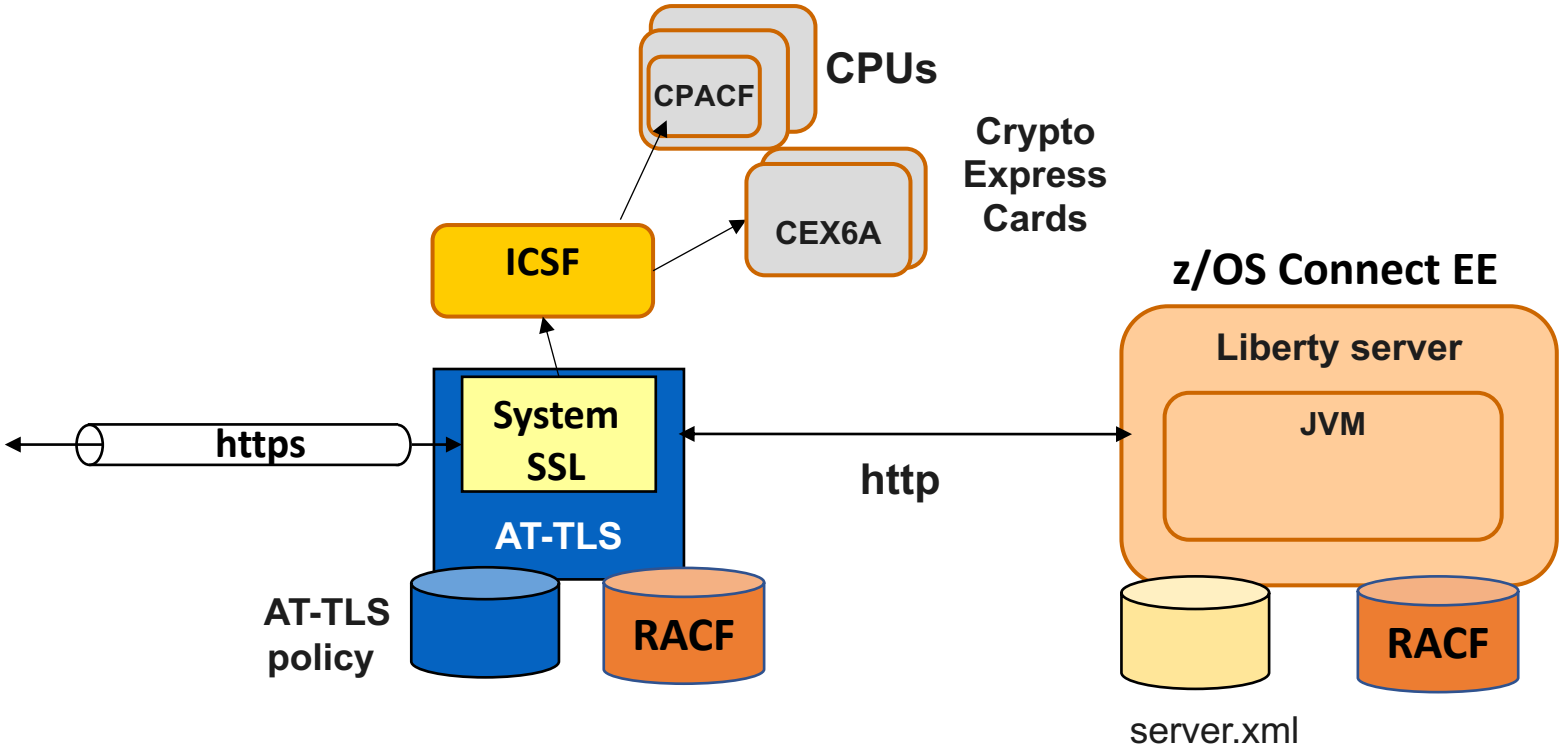
```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Consuming ClientHello handshake message (
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Consumed extension: supported_versions
.java:1168|Negotiated protocol version: TLSv1.2
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Produced ServerHello handshake message (
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Produced server Certificate handshake message (
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Produced ECDH ServerKeyExchange handshake message (
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Produced ServerHelloDone handshake message (
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Consuming ECDHE ClientKeyExchange handshake message (
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Consuming ChangeCipherSpec message
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Consuming client Finished handshake message (
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.java:1168|Produced ChangeCipherSpec message
.java:1168|Produced server Finished handshake message (
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

For more details, see URL https://www.ibm.com/docs/en/sdk-java-technology/8?topic=troubleshooting-debugging-utilities

John.J.Brefach@ibm.com

Using AT-TLS with Liberty

The server XML configuration uses no HTTPS protocol, key rings or other JSSE attributes

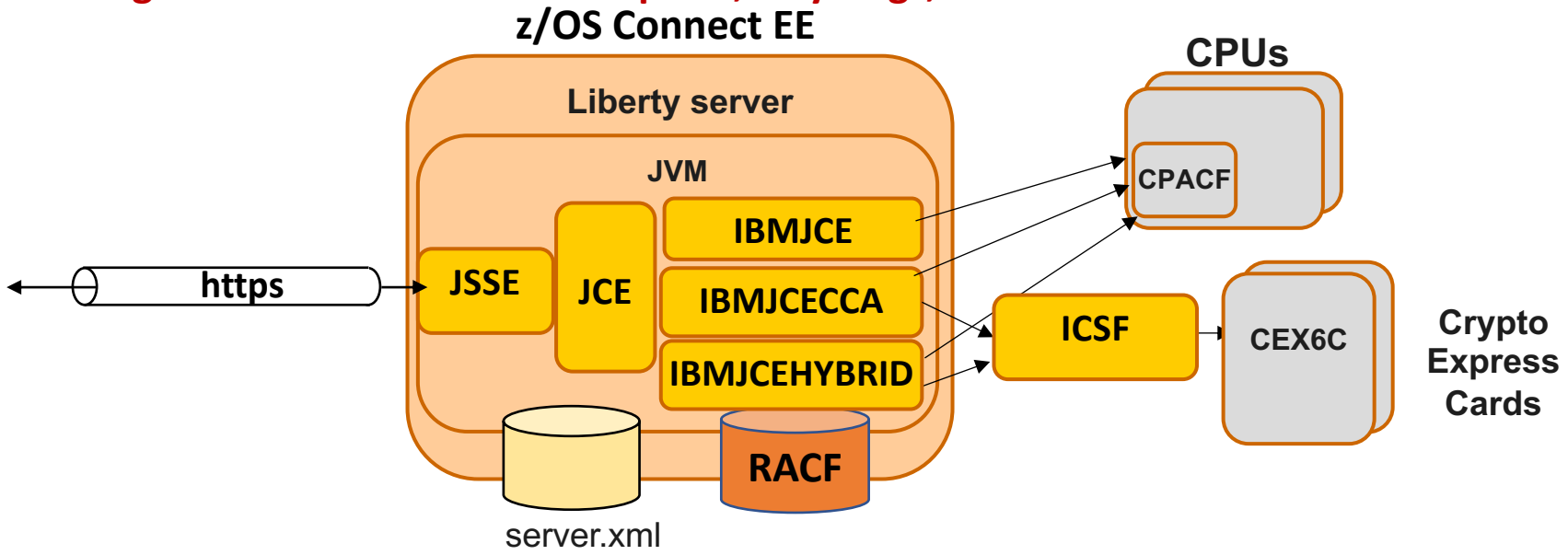


- **Application Transparent TLS** (AT-TLS) creates a secure session on behalf of z/OS Connect
- Only define http ports in server.xml (z/OS Connect does not know that TLS session exists)
- Define TLS protection for all applications (including z/OS Connect) in **AT-TLS policy**
- AT-TLS uses **System SSL** which exploits the CPACF and Crypto Express cards via ICSF

John.J.Brefach@ibm.com

Using JSSE with Liberty

The server XML configuration defines the HTTPS ports, key rings, and other JSSE attributes



- z/OS Connect EE support for TLS is based on **Liberty** server support
- **Java Secure Socket Extension** (JSSE) API provides framework and Java implementation of TLS protocols used by Liberty HTTPS support
- **Java Cryptography Extension** (JCE) is standard extension to the Java Platform that provides implementation for cryptographic services
- **IBM Java SDK** for z/OS provides three different JCE providers, **IBMJCE, IBMJCECCA and IBMJCEHYBRID**.
- The JCE providers access CPACF (**CP Assist for Cryptographic Functions)** directly, therefore keep your Java service levels current.

John.J.Brefach@ibm.com

Other common TLS handshake issues

- **Error occurred during a read, exception:javax.net.ssl.SSLHandshakeException: null cert chain**
  - This exception occurs when the server configuration set to require client certificates (clientAuthenication="true") and the client had no certificate to provide and no alternative authentication method was available.
- **Error occurred during a read, exception:javax.net.ssl.SSLException: Received fatal alert: bad_certificate error (handshake), vc=1083934466 Caught exception during unwrap, javax.net.ssl.SSLException: Received fatal alert: bad_certificate**
  - This is usually caused when the client certificate presented to the server did not have a certificate authority(CA) certificate for the CA that signed the client's personal certificate in the server's trust store key ring.
- **CWWKO0801E: Unable to initialize SSL connection. Unauthorized access was denied or security settings have expired. Exception is javax.net.ssl.SSLHandshakeException: no cipher suites in common**
  - There may be many causes for this issue but first confirm the RACF identity under which the server is running has either READ access to FACILITY resources IRR.DIGTCERT.LISTRING and IRR.DIGTCERT.LIST or access to RDATALIB resources if virtual keyrings are being used.
    - The first FACILITY resource gives the identity access to their own key ring and the second allows access to the certificates. Of if virtual keyrings are in use, then the identity needs READ or UPDATE authority to the <ringOwner>.<ringName>.LST resource in the RDATALIB class. READ access enables retrieving one's own private key, UPDATE access enables retrieving another's private key.
    - An alternative cause: For a TLS handshake to occur, the server must first have access to a private or site certificate that has a private key and the server must have access to that certificate's private key and no certificate with a private key is available.
  - Another possibility is that the TLS handshake the negotiations between the client and server failed, e.g., javax.net.ssl.SSLHandshakeException: Client requested protocol SSLv3 is not enabled or supported in server context

John.J.Brefach@ibm.com

Additional Problem Tracing References

- AT-TL (Application Transparent Transport Layer Security)
  - https://www.ibm.com/docs/en/zos/2.3.0?topic=reference-application-transparent-transport-layer-security-tls
  - https://www.ibm.com/docs/en/zos/2.2.0?topic=traces-sample-tls-trace
  - https://www.ibm.com/docs/en/zos/2.1.0?topic=tls-steps-diagnosing-problems
  - https://www.ibm.com/docs/en/zos/2.2.0?topic=tls-traces#tlstrace__tracelevel
- Liberty
  - https://www.ibm.com/support/pages/set-trace-and-get-full-dump-websphere-liberty
  - https://openliberty.io/docs/latest/log-trace-configuration.html
  - https://www.ibm.com/docs/en/was-liberty/base?topic=liberty-logging-trace
  - https://www.ibm.com/support/pages/mustgather-read-first-websphere-application-server-and-liberty
- Global Services Kit (GSK)
  - https://www.ibm.com/docs/en/developer-for-zos/9.5.1?topic=issues-gsk-ssl-trace
  - https://www.ibm.com/docs/en/itcam-app-mgr/7.2.0?topic=tl-setting-traces-7
- Global Security Toolkit
  - https://www.ibm.com/docs/en/spectrum-protect/8.1.9?topic=codes-global-security-kit-return
  - https://www.ibm.com/docs/en/zos/2.3.0?topic=information-capturing-component-trace-data
  - https://www.ibm.com/docs/en/tivoli-monitoring/6.3.0?topic=options-global-security-toolkit
- Java
  - https://www.ibm.com/support/pages/how-identify-ssltls-keystore-and-truststore-being-used-java%E2%84%A2-application
  - https://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/ReadDebug.html

John.J.Brefach@ibm.com

Protocol Errors during Handshake References

- Problem: try to establish a handshake and these are the errors that occurred
  - https://www.ibm.com/docs/en/ibm-http-server/8.5.5?topic=start-handshake-messages
- Problem: SSLHandshakeException when trying to establish SSL connection with the SMTP server (The application is unable to establish secure communications with the SMTP server after enabling STARTTLS in the mail configuration in IBM Jazz Team Server (JTS). The error in jts.log indicates it cannot convert socket to TLS.)
  - https://www.ibm.com/support/pages/sslhandshakeexception-when-trying-establish-ssl-connection-smtp-server
- https://www.ibm.com/support/pages/ibm-mq-troubleshooting-common-tls-ssl-errors
- https://www.ibm.com/docs/en/ibm-mq/7.5?topic=problems-tlssl-troubleshooting-information

John.J.Brefach@ibm.com

Additional Information

- Anatomy of a Digital certificate
  - https://sites.google.com/site/ddmwsst/digital-certificates#TOC-Anatomy-of-a-Certificate
- https://www.ibm.com/docs/en/ztpf/1.1.0.14?topic=concepts-public-key-cryptography
  - https://www.ibm.com/docs/en/ztpf/1.1.0.14?topic=concepts-symmetric-cryptography

John.J.Brefach@ibm.com

Questions?

Thank you for listening.

Washington Systems Center

John J. Brefach

Washington Systems Center

z/OS Connect SME: Mitch Johnson

mitchj@us.ibm.com