JavaTM Security on z/OS: An Introduction (SHARE Session 1775)

Tom Benjamin
IBM z/OS Java Security Development
tbenjami@us.ibm.com

Java and JVM are Trade marks of Sun Microsystems

What is Java Security?

- Java 2 framework - Set of common cross platform programming API's administered by Sun

- Java Security Extensions - Set of common API's to extend Java 2 to add Security capabilities

- Provides Java Applications easy access to complex Security capabilities within Java framework

- Java Security extensions being added to base Java 2 framework with SDK 1.4.0

z/OS Java SDK with Security components

- IBM Developer kit for OS/390, Java 2 Technology Edition at SDK 1.3.1
  - Available via SMP/E through Boulder
  - Web downloadable http://www-1.ibm.com/servers/eserver/zseries/software/java/

- Made available on z/OS (OS/390) - October 2001
  - Adds 5 new Security components in addition to JAAS and SAF (RACF) interfaces shipped with SDK 1.3.0

- Related Technical article
  - Java security on z/OS, an introduction
    - http://www-1.ibm.com/servers/esdd/articles/jsecurity.html

z/OS Java Security components

- **JAAS** - Java **Authentication** and **Authorization** Service
- **JCE** - Java **Cryptographic** Extension
- **JCE4758 -** Java **Cryptographic** Extension using CCA **hardware cryptographic** devices on z/OS
- **JSSE** - Java **Secure Sockets** Extension (SSL and TLS)
- **CertPath - Certificate** (generation and path validation)
- **PKCS** - **Public Key Standards**
- SAF Interfaces

z/OS
Java Authentication and Authorization Service - JAAS

- **Sun's Java Authentication and Authorization Services (JAAS) framework was released with JDK 1.3.0**
  - **Extends from Java 2 code source based Security model**

- **IBM's z/OS implementation adds support for Principal (userid) based security**
  - **Authentication of a SAF user**
  - **Java Authorization by code source and user**
  - **Based on grants in java.policy file**

- **Documentation available at http://www-1.ibm.com/servers/eserver/zseries/software/java/jaas.html**
- **Related Technical Article**
  - **All that JAAS: An overview of the Java authentication and authorization services**
    - **http://service2.boulder.ibm.com/devtools/news0300/artpag28.htm**

z/OS
Java Authentication and Authorization Service

- **OS/390 Login - User authentication via SAF**
  - ► **User authentication via SAF**
  - ► **Active authentication - Regular password based authentication**
  - ► **Passive authentication - Form Java Principal construct from current z/OS userid associated with the thread of execution**
  - ► **Authorization within Java doas loop**

- **ThreadSubject.doas**
  - ► **Authorization within doas loop and**
  - ► **Change the identity of the underlying z/OS thread within doas loop**

- **SAFPermission**
  - ► **Extend Java permission to use SAF Interfaces**
  - ► **New Java permission to allow Java applications to do authorization checks with SAF for SAF protected resources**

z/OS
Java Cryptography Extension - IBMJCE

- **Implements platform independent Cryptography API into Java 2 as a standard extension**
  - **Cryptography is performed via software**
- **Replaces IBMJCA capabilities**
  - **Digital Signatures, Hashing, keystore**
  - **Extends to add more capabilities**
- **Includes many algorithms for**
  - **Encryption/Decryption (Symmetric and Asymmetric algorithms)**
  - **Key agreement, MAC**
- **Code is common with other IBM platforms at SDK 1.3.1 level**
- **Documentation available at**
  http://www-1.ibm.com/servers/eserver/zseries/software/java/jce.html

z/OS
Java Cryptography Extension - IBMJCE

- Digital Signatures via RSA and DSA
- Hashing - SHA1, MD2, MD5
- Keystore - Symmetric and Asymmetric keys protected by 3DES
- Symmetric Algorithms - DES, 3DES, PBE, Blowfish, Mars, RC2, RC4
    - Ciphers - ECB, CBC, CFB, OFB, PCBC
- Asymmetric Algorithms - RSA
- Key Agreement - Diffie-Hellman
- HMAC - MD5, SHA1

z/OS
Java Cryptography Extension - IBMJCE
A simple code example - DES

```
// generate the DES key
        java.security.SecureRandom random =
java.security.SecureRandom.getInstance("IBMSecureRandom");
        SecretKey key = null;
        KeyGenerator desKeyGen;
        try {
          // take the first DES in the provider list java.security
          desKeyGen = KeyGenerator.getInstance("DES");
        } catch (Exception ex) {
          System.out.println("Unexpected exception1: " + ex.getMessage());
          return;
        }
        try {
          desKeyGen.init(random);
          key = desKeyGen.generateKey();
        } catch (Exception ex) {
          System.out.println("Unexpected exception2: " + ex.getMessage());
          return;
        }
```

z/OS
Java Cryptography Extension - IBMJCE
A simple code example DES cont.

```
 // Create the Cipher and encrypt code here
try {
  // take the first provider in the provider list with DES/CBC/PKCS5Padding
     cp = Cipher.getInstance("DES/CBC/PKCS5Padding");
     cp.init(Cipher.ENCRYPT_MODE, key);
         cipherText1 = cp.update(byteDataToCipher);
         cipherText2 = cp.doFinal();
 } catch (Exception e) {
     System.out.println("Exception hit ==> "+e);
 }
```

z/OS
Java Cryptography Extension - IBMJCE

- **Related Technical articles**
  - **Java cryptography Part 1: Encryption and decryption**
    - **http://service2.boulder.ibm.com/devtools/news0100/artpage18.htm**
  - **Java Cryptography Part II: Key generation and management**
    - **http://service2.boulder.ibm.com/devtools/news0300/artpag20.htm**
  - **Java cryptography Part III: Implementing your own provider**
    - **http://service2.boulder.ibm.com/devtools/news0600/art19.htm**
  - **Java Cryptography Part IV: JCE export considerations**
    - **http://service2.boulder.ibm.com/devtools/news0900/art5.htm**

z/OS
Java Cryptography Extension - IBMJCE4758

- IBM Implementation of JCE Cryptography using CCA hardware cryptographic devices
- Replaces those JCE capabilities available via CCA hardware
- No changes to the JCE API's
  - Software cryptography replaced by calls made to IBM's CCA hardware inside the provider
- Almost no changes to Java JCE Applications
  - key generation
  - java.security (properties file) provider order
- Allows a JCE application to take advantage of hardware cryptography without extensive knowledge of hardware cryptography
- Documentation available at http://www-1.ibm.com/servers/eserver/zseries/software/java/jcecca.html

z/OS
Java Cryptography Extension - IBMJCE4758

- Greatly enhances security
  - Cryptographic processing done via secure devices
  - Adds Protected keys (never available in the clear)
  - Adds Retained keys (stored on the hardware cryptographic device and never available in the clear)

- Greatly enhances performance
  - Digital Sign/Verify as much as 34 times faster than software cryptography
  - Moves Cryptographic operations off the CPU and onto the hardware cryptographic device
  - Faster throughput and a reduction in CPU usage

z/OS
Java Cryptography Extension - IBMJCE4758

Sign and Verification	Software CLEAR keys		Hardware CLEAR keys		Hardware PKDS keys		Hardware RETAINED keys	
Algorithm and hash used	Trans per second	CPU utilization	Trans per second	CPU utilization	Trans per second	CPU utilization	Trans per second	CPU utilization
RSA MD2	67	97.82%	1,018	81.84%	1,033	83.35%	99	7.87%
RSA MD5	68	97.20%	880	35.57%	790	33.95%	98	3.77%
RSA SHA1	67	97.09%	907	41.26%	812	39.84%	98	4.02%
DSA SHA1	123	90.59%	---N/A--	---N/A----	361	12.69%	---N/A--	----N/A----

2064-116 (16 CPU's) with 2 CCF's, 16 IBM 4758-2 Cryptographic Coprocessor PCI cards
running z/OS V1R2, Java 1.3.1 (SDK 1.3.1 level (PTF UQ99325) IBMJCE (software cryptographic provider)
and IBMJCE4758 (Hardware cryptographic provider). The data size was 1024, with 50 threads of execution

z/OS

Java Cryptography Extension - IBMJCE4758

Encryption	**Software cryptography IBMJCE**			**Hardware cryptography IBMJCE4758**		
Algorithm and data size	ETR Trans per second	CPU Utilization	ITR Trans Per second	ETR Trans per second	CPU Utilization	ITR Trans Per second
DES 1KB	5,401	96.02%	5,625	5,186	26.95%	19,243
DES 100KB	89	95.31%	94	362	22.44%	1,613
DES 1mb	2	97.78%	9	64	15.26%	419
Triple DES 1KB	3,143	99.94%	3,145	5,159	29.62%	17,417

2064-116 (16 CPU's) with 2 CCF's, 16 IBM 4758-2 Cryptographic Coprocessor PCI cards
running z/OS V1R2, Java 1.3.1 (SDK 1.3.1 level (PTF UQ99325) IBMJCE (software cryptographic provider)
and IBMJCE4758 (Hardware cryptographic provider). The data size was 1024, with 50 threads of execution

z/OS
Java Cryptography Extension - IBMJCE4758

- **Digital Signatures via RSA and DSA**

- **Hashing - SHA1, MD2, MD5**

- **Keystore - Symmetric and Asymmetric keys protected by 3DES**

- **Symmetric Algorithms - DES, 3DES, PBE**
  - **Ciphers - ECB, CBC, CFB, OFB, PCBC**

- **Asymmetric Algorithms - RSA**

- **HMAC - MD5, SHA1**

z/OS
Java Cryptography Extension - IBMJCE4758
A simple code example - DES

Nothing changes from the IBMJCE example

```
// generate the DES key
        java.security.SecureRandom random =
java.security.SecureRandom.getInstance("IBMSecureRandom");
        SecretKey key = null;
        KeyGenerator desKeyGen;
        try {
           // take the first DES in the provider list java.security
           desKeyGen = KeyGenerator.getInstance("DES");
        } catch (Exception ex) {
           System.out.println("Unexpected exception1: " + ex.getMessage());
           return;
        }
        try {
           desKeyGen.init(random);
           key = desKeyGen.generateKey();
        } catch (Exception ex) {
           System.out.println("Unexpected exception2: " + ex.getMessage());
           return;
        }
```

z/OS
Java Cryptography Extension - IBMJCE4758
A simple code example DES cont.

Nothing changes from the IBMJCE example

```
// Create the Cipher and encrypt code here
    try {
      // take the first provider in the provider list with DES/CBC/PKCS5Padding
         cp = Cipher.getInstance("DES/CBC/PKCS5Padding");
         cp.init(Cipher.ENCRYPT_MODE, key);
             cipherText1 = cp.update(byteDataToCipher);
             cipherText2 = cp.doFinal();
     } catch (Exception e) {
         System.out.println("Exception hit ==> "+e);
     }
```

Much better examples in the technical articles referenced later

z/OS
Java Cryptography Extension - IBMJCE4758

- **Related Technical articles**
  - **Java Cryptography Architecture using Hardware cryptography -- part 1, an introduction**
    - **http://www-1.ibm.com/servers/esdd/articles/java_crypto.html**
  - **Java Cryptography Architecture using Hardware cryptography -- part 2, details for z/OS**
    - **http://www-1.ibm.com/servers/esdd/articles/java_crypto2.html**
  - **Java Cryptography Extension using hardware cryptography -- part 3**
    - **http://www-1.ibm.com/servers/esdd/articles/java_crypto3.html**
  - **More coming at IBM eServer Developer Domain**
    - **http://www-1.ibm.com/servers/esdd/index.html**

z/OS
Java Secure Sockets Extension - IBMJSSE

- Implements SSL 3.0 and TLS 1.0 as Java2 standard extensions
  - 100% pure Java Implementation
- Provides Authentication, Integrity and Privacy at the transport level
  - privacy for browser to Web-Server e-business
  - any secure data exchange
- Supports common security algorithms
  - RSA, DSA, DES, 3DES
- Socket factories encapsulate socket creation, key and trust management behavior for ease of use
- Code is common with other IBM platforms at SDK 1.3.1 level
  - Allows for application portability
- Documentation available at http://www-1.ibm.com/servers/eserver/zseries/software/java/jsse.html

z/OS
Java Secure Sockets Extension - IBMJSSE

- Advantages of IBMJSSE
  - Supports a wide variety of SSL and TLS algorithm types
  - Easier Socket Creation via encapsulated factories
  - Ability to create application specific Trust Manager for application requirements

- IBMJSSE is the preferred SSL/TLS for Java Applications on z/OS
  - IBMJSSE is 100% pure Java and does not use System SSL services
  - IBMJSSE should be used in place of System SSL for Java Applications
    - No overhead converting to C based services (JNI)

z/OS
Java Secure Sockets Extension - IBMJSSE

- Algorithms for key exchange and authentication
  - RSA, Diffie-Hellman, DSA
- Algorithms for Data exchange
  - DES, 3DES, RC4, RC2
- Hashing Algorithms
  - SHA, MD5

z/OS
Java Secure Sockets Extension - IBMJSSE

- **Cipher Suites supported**
  - SSL_RSA_WITH_RC4_128_MD5
  - SSL_RSA_WITH_RC4_128_SHA
  - SSL_RSA_WITH_DES_CBC_SHA
  - SSL_RSA_WITH_3DES_EDE_CBC_SHA
  - SSL_DHE_RSA_WITH_DES_CBC_SHA
  - SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
  - SSL_DHE_DSS_WITH_DES_CBC_SHA
  - SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
  - SSL_RSA_EXPORT_WITH_RC4_40_MD5
  - SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
  - SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
  - SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
  - SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
  - SSL_RSA_WITH_NULL_MD5
  - SSL_RSA_WITH_NULL_SHA
  - SSL_DH_anon_WITH_RC4_128_MD5
  - SSL_DH_anon_WITH_DES_CBC_SHA
  - SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
  - SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
  - SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- **Also available for TLS**

z/OS
Java Secure Sockets Extension - IBMJSSE
A simple code example - Client Side

```
 // Makes an SSLSocketFactory  - Use all defaults for handshake and privacy type
    socketFactory = SSLSocketFactory.getDefault();

 // Use socketFactory to create a socket
  socket = socketFactory.createSocket(
    InetAddress.getLocalHost(), port);

// get input and output stream from the socket for the client
    dos = new DataOutputStream(socket.getOutputStream());
    dis = new DataInputStream(socket.getInputStream());

// send some text
    dos.writeUTF(text);
```

Much better examples in the technical articles referenced

z/OS
Java Secure Sockets Extension - IBMJSSE

- **Related Technical Articles**
  - Exploiting SSL in Java
    - http://service2.boulder.ibm.com/devtools/news0800/art37.htm
  - Exploiting SSL in Java Security: A reprise
    - http://service2.boulder.ibm.com/devtools/news0900/art8.htm
  - Can I trust my Java Secure Sockets Extension provider?
    - http://www.developer.ibm.com/library/articles/programmer/trust.html

z/OS
Java Certification Path - CertPath

- Set of classes and interfaces to create, build and validate digital certification paths
- Compliant with 8th version of the Internet draft for PKI Certificate and CRL Profile (PKIX)
- Support for LDAP and Collection CertStores
- Usage - Designing secure applications that build or validate certification paths
- 100% pure Java implementation

- Documentation available at http://www-1.ibm.com/servers/eserver/zseries/software/java/certpath.html
- Related Technical article
  - ► Certification paths Weaving a web of trust for e-business
    - – http://www-106.ibm.com/developerworks/library/it-certpath/?dwzone=ibm

z/OS
Java Certification Path - CertPath

- Based on the Java Cryptographic Service Provider architecture
- General CertPath capabilities:
  - CertificateFactory: X.509 CertPath type with PKCS7 and PkiPath encodings
  - CertPathValidator: Validate the Certificate path via PKIX algorithm
  - CertPathBuilder: Builds a certificate path via PKIX algorithm
  - CertStore: Certificate collections - LDAP and other certificate stores

z/OS
Public Key Cryptographic Standards - IBMPKCS

- PKCS - Set of de-facto standards widely used for Public Key Cryptography
- IBMPKCS - IBM's Set of Java classes that provide access / usage of several of these standards
  - PKCS 1 - RSA Cryptography
  - PKCS 5 - Password-Based Encryption
  - PKCS7 - Cryptographic Message Syntax
  - PKCS8 - Private-Key Information Syntax
  - PKCS9 - Selected Attribute types
  - PKCS10 - Certificate Request Syntax
  - PKCS12 - Personal Information Exchange Syntax
  - S/MIME - Secure Multipurpose Mail Extensions
- Documentation available at http://www-1.ibm.com/servers/eserver/zseries/software/java/cryptstan.html

z/OS
Public Key Cryptographic Standards - IBMPKCS

- **Provides Java applications the ability to use the PKCS standards**
- **Also Makes the S/MIME standards available to Java applications**
  - **S/MIME capabilities require a cryptographic provider like IBMJCE**
- **IBMPKCS is also used by several of the earlier Java Security components**
  - **IBMJCE**
  - **IBMJCE4758**
  - **CertPath**
- **Good example of how the Java Security components build on each other**

z/OS
SAF Interfaces

- Java static class methods provide an interface to the z/OS Security Server using SAF (Secure Architecture Facility) and z/OS services to provide basic authentication and authorization services.
  - PlatformSecurityServer class
    - IsActive(), resourceIsActive()
  - PlatformUser class
    - authenticate(), changePassword(), isUserInGroup()
  - PlatformAccessControl.checkPermission()
  - PlatformThread.getUserName()
- Documentation available at http://www-1.ibm.com/servers/eserver/zseries/software/java/security.html

z/OS Java SDK with
Security components - Summary

- **IBM Developer kit for OS/390, Java 2 Technology Edition at SDK 1.3.1**
  - **Adds 5 new Security components in addition to JAAS and SAF interfaces shipped with SDK 1.3.0**
    - **IBMJCE** - Java **Cryptographic** Extension
    - **IBMJCE4758 -** Java **Cryptographic** Extension using CCA **hardware cryptographic** devices
    - **IBMJSSE -** Java **Secure Sockets** Extension (SSL and TLS)
    - **CertPath** - **Certificate** (generation and validation)
    - **IBMPKCS - Public Key Standards**
  - **Set of common API's to extend Java 2 Security capabilities**
  - **Provides Java Applications easy access to complex Security capabilities within Java framework on z/OS**

Quiz ???

JAAS, JCE, JCE4758, JSSE, SSL, TLS, CertPath, PKCS, SAF, CCA, JCA, MAC, RSA, DSA, SHA1, DES, 3DES, MD2, MD5, PBE, Blowfish, Mars, RC2, RC4, ECB, CBC, CFB, OFB, PCBC, HMAC, ETR, ITR, PKIX, CRL, PKI, X.509, S/MIME, RACF