z/OS Java™ Security

Brian Beegle
IBM Corp.
beegle@us.ibm.com

February 25, 2008
Session 5563

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

IBM®
RACF®
System z9
Z800
Z880
Z9
Z900
Z990
z/OS®

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

All other products may be trademarks or registered trademarks of their respective companies.

Notes:

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

What is Java Security?

- Java framework - Set of common cross platform programming APIs in Java Platform, Standard Edition (Java SE) administered by Sun
- Java Security Extensions - Set of common APIs to extend Java to add Security capabilities
- Provides Java Applications easy access to complex Security capabilities within Java framework
- Java Security extensions were integrated into base Java 2 (J2SE) framework in SDK 1.4.0 (available since 2Q2002)

Java Security Service Provider Architecture Bird's Eye View

- Service Classes (engines)
  - represent cryptographic and other security-related services (operations)
    - Cipher, KeyAgreement, KeyGenerator, Mac, SecretKeyFactory
  - abstract in nature
  - Service Provider Interface (SPI)
    - methods which must be implemented (also abstract)
  - Application Programming Interface (API)
    - public methods defined by engine needed to use services
- Provider architecture
  - supplies algorithm implementations for service classes
  - collection of all implementations referred to as "service provider" or simply "provider"
  - supports the development of plug-replaceable components (providers)
- More information at

http://java.sun.com/j2se/1.5.0/docs/guide/security/HowToImplAProvider.html

Java Security Service Provider Architecture *(continued)*

- Provider configuration determines in which provider desired algorithm is found, on call to getInstance()
- Two ways to select provider containing algorithm:
  - if no specific provider is designated on the getInstance() call, the providers are checked in the provider list order (specified in java.security configuration file), selecting the first provider which implements the algorithm.
    - Cipher c1 = Cipher.getInstance("DES")
  - Location of java.security file
    - default: ${java-home}/lib/security/java.security:
      security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
      - security.provider.2=com.ibm.crypto.provider.IBMJCE
    - JVM startup parameter: java -Djava.security.properties=/my.user/java.security

Java Security Service Provider Architecture *(continued)*

- an explicit getInstance() call to create a cipher instance with a particular provider will select algorithm for cipher from designated provider, if it implements it (throws exception, otherwise)
  - Cipher c1 = Cipher.getInstance("DES", "IBMJCECCA")
    c1.init(Cipher.ENCRYPT_MODE, myKey) (SDK 5 and later)

- Delayed provider selection introduced in SDK 5
  - provider selection made during relevant initialization method
  - ensures provider can use specified key object
  - looks for first provider which satisfies engine/algorithm/key type support

IBM Java Security Providers

- IBMJCE - Java Cryptographic Extension
- IBMJCECCA - JCE using z/OS hardware crypto (z/OS only)
- IBMJSSE2 - Java Secure Sockets Extension (SSL & TLS)
- IBMPKCS11Impl - Public Key Cryptographic Standard #11
- IBMJAAS - Java Authentication and Authorization Service
- IBMJGSS - Generic Security Services - Kerberos, GSS-API
- IBMCERTPATH - Certificate Path Validation
- IBMSASL - Simple Authentication & Security Layer
- IBMXMLCryptoProvider - XML Digital Signatures
- IBMXMLEncProvider - XML Digital Encryption
- SAF Interfaces - System Authorization Facility (z/OS only)

IBM Java Cryptography Extension - IBMJCE

- Implements platform-independent Cryptography API into Java 2 as a standard extension (JDK 1.4)
  - Cryptography is performed via software
- Replaces IBMJCA (Java Cryptographic Architecture) capabilities (from 1.3.0 JDK level)
  - Digital Signatures, Hashing, keystore
  - Extends to add more capabilities
- Includes many algorithms for
  - Encryption/Decryption (Symmetric and Asymmetric algorithms), Sign/Verify
  - Key agreement, Message Authentication Code (MAC)

IBM Java Cryptography Extension - IBMJCE *(continued)*

- Digital Signatures via RSA and DSA (Digital Signature Algorithm)
- Hashing - SHA1 (Secure Hash Algorithm), MD2 (Message Digest), MD5
- Keystore - Symmetric and Asymmetric keys protected by 3DES
- Symmetric Algorithms - DES (Data Encryption Standard), 3DES, AES (Advanced Encryption Standard), PBE (Password Based Encryption), Blowfish, Mars, RC2, RC4
  - Cipher Modes - Electronic Code Block (ECB), Cipher Block Chaining (CBC), Cipher Feedback Mode (CFB), Output Feedback Mode (OFB), Propagating Cipher Block Chaining (PCBC)
- Asymmetric Algorithms - RSA (Rivest-Shamir-Adelman)
- Key Agreement - Diffie-Hellman
- Hash-based Message Authentication Code (HMAC) - MD5, SHA1

IBM Java Cryptography Extension - IBMJCE *(continued)*

- IBMJCE z/OS platform-specific
  - Same code and capabilities as other IBM platforms
  - Adds the capability to use SAF-based keys/certificates (any external security managers implementing SAF interfaces, such as RACF)
    - keystore for SAF Digital Certificate (key ring) Support
    - certificates always accompanied by public key and optionally accompanied by a private key to create an asymmetric key pair
    - no symmetric key support
- Code is common on IBM platforms at SDK 1.4.2 level & above
- Granted Cryptographic Module Validation Program (CMVP) FIPS 140-2 level 1 certification
  - Separate FIPS-only provider (IBMJCEFIPS)
  - http://cs-www.ncsl.nist.gov/cryptval/

IBM Java Cryptography Extension - IBMJCE *(continued)*

- New in SDK 5:
  - RSA with Optimal Asymmetric Encryption Padding (OAEP) added
  - SHA2withRSA, SHA3withRSA and SHA5withRSA signature algorithms added
  - HmacSHA2, HmacSHA3, HmacSH5 MAC algorithms added
  - ElGamal cipher added
- New in SDK 6:
  - CipherText Stealing mode
  - ISO-10126 Padding support
  - PBKDF2HmacSHA1Factory and PBKDF2KeyImpl (PKCS#5)
  - Added supporting classes for TLS use with JSSE2

Java Security Cryptographic Strength

- Full Function versus Limited Key Size Cryptography
  - default: US_export_policy.jar and local_policy.jar pre-installed in directory ${java-home}/lib/security: limited function cryptography with no export restrictions
  - download new files from http://www-03.ibm.com/servers/eserver/zseries/software/java/j6jcecca.html
    - replace files US_export_policy.jar and local_policy.jar in ${java-home}/lib/security: full function cryptography
    - also available in ${java-home}/demo/jce/policy-files/unrestricted

z/OS Java Cryptography Extension - IBMJCECCA

- IBM Implementation of JCE Cryptography using z/OS Common Cryptographic Architecture (CCA) hardware cryptographic devices
- Replaces those JCE capabilities available via CCA hardware
- 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
  - Uses Integrated Cryptographic Services Facility (ICSF) for hardware interfaces
  - Requires ICSF installation and initiation
  - Remember to activate hardware

z/OS Java Cryptography Extension - IBMJCECCA *(continued)*

- Enhances security
  - Cryptographic processing done via secure devices
  - Adds Protected (secure) keys (never available in the clear when in use or storage)
  - Can be used for key management(sign/verify & encrypt/decrypt) OR signature(sign/verify only)
- Exploits performance of hardware
  - Moves Cryptographic operations off the CPU and onto the hardware cryptographic device

z/OS Java Cryptography Extension - IBMJCECCA *(continued)*

- Types of keys:
  - clear - is stored as an external hardware key structure or token (clear text is tokenized into a CCA external token)
    - greatest throughput
    - lowest hardware security
  - PKDS (Public/private Key Data Set) - private key is encrypted with the system master key so that the clear text version of this key can never be viewed or retrieved (secure key); key pair is stored in a system key storage area
    - moderate hardware security
  - CKDS (Cryptographic Key Data Set) - stores operational keys of all types (except PKA keys), each encrypted with system master key, similar to PKDS above
  - More info at http://www.ibm.com/developerworks/eserver/articles/java_crypto.html

z/OS Java Cryptography Extension - IBMJCECCA *(continued)*

SHARE
Technology · Connections · Results

- Digital Signatures via RSA and DSA (DSA on z900/z800 only)
- Hashing - SHA1, SHA-256, MD2, MD5
- Keystore - Symmetric and Asymmetric keys protected by 3DES
- Symmetric Algorithms - DES, 3DES, PBE, AES
  - Cipher Modes - ECB, CBC, CFB, OFB, PCBC
- Asymmetric Algorithms - RSA
- HMAC - MD5, SHA1

- Adds the capability to use SAF based keys/certificates (RACF)
  - keystore for SAF Digital Certificate (key ring) Support

z/OS Java Cryptography Extension - IBMJCECCA *(continued)*

SHARE
Technology · Connections · Results

- New in SDK 5:
  - Name change to IBMJCECCA, reflecting type of hardware interface
    - Aliasing to IBMJCE4758 for backward compatibility
  - Support for PKCS#12 export of CLEAR keys saved in a JCE4758KS/JCECCAKS keystore
  - IBMJCECCA performance improvements for certain operations implemented on the CPACF (DES, TDES, SHA-1)
  - Support for the following: AES, AES/CBC for both NoPadding and PKCS5Padding, AES/ECB for both NoPadding and PKCS5Padding, DES/ECB HW support, and DESede HW Support for both NoPadding and PKCS5Padding
  - Supports access to keys via ICSF key label; gives the ability to use pre-existing PKDS and CKDS entries for doing various JCE operations
  - Support for encrypted (secure) symmetric key wrapping and encrypted (secure) symmetric key unwrapping

z/OS Java Cryptography Extension - IBMJCECCA *(continued)*

- New in SDK 5 (continued):
  - Key wrapping of clear hardware symmetric keys for import and export of keys
  - CFB mode for AES, 3DES, DES
  - Key token access for secure keys

z/OS Java Cryptography Extension - IBMJCECCA *(continued)*

SHARE
Technology · Connections · Results

- New in SDK 6:
  - Writeable RACF keystores
    - operations previously available only through RACDCERT
    - write access via APIs to RACF keyrings through the JCECCARACFKS and JCERACFKS KeyStores
  - Mixed-case key label alias support
    - Java implementations have always created and used key aliases in lower-case form
    - TSO-based utilities permit creation of RACF key aliases with mixed-case labels, so it is possible to create RACF keys through utilities which cannot be referenced using standard Java APIs
    - Can now use keys which were created with mixed-case (which includes upper-case) label aliases for RACF keystores ONLY
      - IBMJCERACFKS and IBMJCECCARACFKS

z/OS Java Cryptography Extension - IBMJCECCA *(continued)*

- Documentation available at http://www-03.ibm.com/servers/eserver/zseries/software/java/j6jcecca.html

(Related Technical articles no longer available as of 22 Feb. 2008.)

IBM Public Key Cryptographic Standard #11 - IBMPKCS11Impl

- Implementation of RSA's Public Key Cryptographic Standard #11
  - New provider introduced in SDK 6
  - Smart card interface standard
  - APIs define cryptographic objects (keys, certificates, etc) and all the functions needed to use, create/generate, modify and delete those objects
  - Supports subset of V2.20 PKCS#11 spec
- Implemented in Java for z/OS using the IBMPKCS11Impl provider
  - Uses JCE and JCA frameworks to add capability to use hardware cryptographic devices through PKCS#11 interfaces
- Requires z/OS V1R9

IBM Public Key Cryptographic Standard #11 - IBMPKCS11Impl *(continued)*

- Supported algorithms:
  - # Signature.MD5withRSA
  - # Signature.SHA1withRSA
  - # Signature.SHA256withRSA
  - # Signature.RSAforSSL
  - # Cipher.RSA/SSL/PKCS1Padding
  - # Cipher.RSA/SSL/NoPadding
  - # Cipher.RSAforSSL
  - # Cipher.DES/CBC/NoPadding
  - # Cipher.DES/CBC/Pad
  - # Cipher.DES/ECB/NoPadding
  - # Cipher.DESede/CBC/NoPadding
  - # Cipher.DESede/CBC/Pad
  - # Cipher.DESede/ECB/NoPadding
  - # Cipher.AES/CBC/NoPadding
  - # Cipher.AES/CBC/Pad
  - # Cipher.AES/ECB/NoPadding

IBM Public Key Cryptographic Standard #11 - IBMPKCS11Impl *(continued)*

- Supported algorithms (continued):
  - # KeyPairGenerator.RSA
  - # KeyGenerator.DES
  - # KeyGenerator.DESede
  - # KeyGenerator.AES
  - # MessageDigest.MD5
  - # MessageDigest.SHA1
  - # MessageDigest.SHA-256
  - # KeyFactory.RSA
  - # SecretKeyFactory.DES
  - # SecretKeyFactory.DESede
  - # SecretKeyFactory.AES

IBM Public Key Cryptographic Standard #11 - IBMPKCS11Impl *(continued)*

- To use the IBMPKCS11Impl provider on the z/OS platform, you must have the following:
- A system at the z/OS V1R9 level with one of the following:
  - On a z800 or z900 processor, a CCF and a PCICC card
  - On a z890 or z990 processor, a CPACF and a PCIXCC card
  - On a z890 or z990 processor, a CPACF and a CEX2C card
  - On a z9 processor, a CPACF and a CEX2C or CEX2A card
- ICSF must be running

- See the z/OS V1R9 Cryptographic Services Integrated Cryptographic Services Facility (ICSF) documentation for a description of the functions available for each of the configurations.

IBM Public Key Cryptographic Standard #11 - IBMPKCS11Impl *(continued)*

- Usage Notes
  - Only clear keys supported in the Token Data Set (TKDS); no secure key support
    - Use IBMJCECCA if secure keys are required
  - No support in IBMPKCS11Impl on z/OS for PKCS#11 hardware exploitation by IBMJSSE2
    - Use IBMJCECCA if hardware exploitation with IBMJSSE2 required

IBM Public Key Cryptographic Standard #11 - IBMPKCS11Impl *(continued)*

- Where to start?
  - Java Public Key Cryptographic Standards #11 Implementation Provider Overview at http://www-03.ibm.com/servers/eserver/zseries/software/java/j6pkcs.html
  - z/OS IBMPKCS11Impl Guide (configuration, initialization, supported algorithms) at http://www-03.ibm.com/servers/eserver/zseries/software/java/j6pkcs11implgd.html
- Additional references:
  - z/OS Cryptographic Services Integrated Cryptographic Services Facility Writing PKCS#11 Application SA23-2231
  - Java Security PKCS#11 Implementation Provider document and IBMPKCS11Impl classes and methods API documentation
    - http://www-128.ibm.com/developerworks/java/jdk/security/60/
  - z/OS Security Server RACF Command Language Reference SA22-7687 and z/OS Security Server RACF Security Administrator's Guide SA22-7683 for information on RACDCERT
  - z/OS Cryptographic Services System Secure Sockets Layer Programming SC24-5901 (for gskkyman information)

IBM Java Secure Sockets Extension - IBMJSSE2

- Implements SSL 3.0 and TLS 1.0 as Java 2 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, AES, 3DES, RC4
- Socket factories encapsulate socket creation, key and trust management behavior for ease of use
- Code is common with other IBM platforms at SDK 5 and above
  - Allows for application portability

IBM Java Secure Sockets Extension - IBMJSSE2 *(continued)*

- Benefits of IBMJSSE2
  - Supports a wide variety of SSL and TLS algorithm types
  - Easier Socket Creation via encapsulated factories
  - Ability to create application specific Trust Manager or Key Manager for application requirements
- IBMJSSE2 is the preferred SSL/TLS for Java Applications on z/OS
  - IBMJSSE2 is 100% pure Java and does not use System SSL services
  - IBMJSSE2 should be used in place of System SSL for Java Applications

IBM Java Secure Sockets Extension - IBMJSSE2 *(continued)*

- Algorithms for key exchange and authentication
  - RSA, Diffie-Hellman, DSA
- Algorithms for Data exchange
  - DES, 3DES, AES, RC4, RC2
- Hashing Algorithms
  - SHA1, MD5

IBM Java Secure Sockets Extension - IBMJSSE2

SHARE
Technology · Connections · Results

- Enhancements in IBMJSSE2 Provider vs. older IBMJSSE Provider:
  - Improved serviceability
    - Improved tracing and debug capability
    - Tracing now configurable - dynamic debug tracing support is accessed with the system property javax.net.debug
  - Uses IBM cryptographic providers
    - IBMJSSE Provider used it own internal cryptographic code
    - IBMJSSE2 Provider uses IBM's JCE providers: IBMJCE, IBMJCEFIPS, and IBMJCECCA
  - IBM's JSSE2 provider no longer needs to be FIPS certified.

IBM Java Secure Sockets Extension - IBMJSSE2 *(continued)*

SHARE
Technology · Connections · Results

- Enhancements in IBMJSSE2 Provider (continued):
  - Supports using JSSE with hardware cryptographic accelerators and hardware keys
    - performance improvements
    - acceleration for handshake and payload
    - hardware cryptographic accelerator exploitation based on key type and provider list order
  - Supports a second, PKIX-compliant TrustManager
    - implemented using the default CertPath PKIX implementation (compliant with RFC 3280)
    - default Trustmanager in SDK 6
    - simple X.509-based TrustManager still available

IBM Java Secure Sockets Extension - IBMJSSE2 *(continued)*

- Cipher Suites supported
  - SSL_RSA_WITH_RC4_128_MD5
  - SSL_RSA_WITH_RC4_128_SHA
  - SSL_RSA_WITH_AES_128_CBC_SHA
  - SSL_RSA_WITH_AES_256_CBC_SHA
  - SSL_RSA_WITH_DES_CBC_SHA
  - SSL_RSA_FIPS_WITH_DES_CBC_SHA
  - SSL_RSA_WITH_3DES_EDE_CBC_SHA
  - SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
  - SSL_DHE_RSA_WITH_AES_128_CBC_SHA
  - SSL_DHE_RSA_WITH_AES_256_CBC_SHA
  - SSL_DHE_RSA_WITH_DES_CBC_SHA
  - SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
  - SSL_DHE_DSS_WITH_AES_128_CBC_SHA
  - SSL_DHE_DSS_WITH_AES_256_CBC_SHA
  - SSL_DHE_DSS_WITH_RC4_128_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

IBM Java Secure Sockets Extension - IBMJSSE2 *(continued)*

- Cipher Suites supported (continued)
  - 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_AES_128_CBC_SHA
  - SSL_DH_anon_WITH_AES_256_CBC_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

IBM Java Secure Sockets Extension - IBMJSSE2 *(continued)*

- Cipher Suites supported (continued)
  - TLS_KRB5_WITH_RC4_128_SHA
  - TLS_KRB5_WITH_RC4_128_MD5
  - TLS_KRB5_WITH_3DES_EDE_CBC_SHA
  - TLS_KRB5_WITH_3DES_EDE_CBC_MD5
  - TLS_KRB5_WITH_DES_CBC_SHA
  - TLS_KRB5_WITH_DES_CBC_MD5
  - TLS_KRB5_EXPORT_WITH_RC4_40_SHA
  - TLS_KRB5_EXPORT_WITH_RC4_40_MD5
  - TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA
  - TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5

IBM Java Secure Sockets Extension - IBMJSSE2 *(continued)*

- New in SDK 5:
  - Now uses the Java Cryptography Extension (JCE) for all its cryptographic algorithms.
  - Now supports pluggable hardware crypto providers
  - SSLEngine (non-blocking I/O) allows SSL/TLS applications to choose their own I/O and compute models.
  - Enhanced TrustManager support.
  - HTTP/HTTPS enhancements (dynamic proxy server config & selection, pluggable caching support, pluggable cookie support, connect and read timeout methods)
  - Kerberos cipher suites are available, if supported by the operating system
- Documentation available at http://www-03.ibm.com/servers/eserver/zseries/software/java/j5jsse.html

IBM
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 implementation adds support for Principal (userid) based security
  - Authentication of a user
  - Java Authorization by code source and user
  - Enforce new access controls based on who has authenticated
  - Based on grants in java.policy file
    - grant permissions to specific principals

z/OS-Specific
Java Authentication and Authorization Service *(continued)*

- z/OS Login
  - 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
- ThreadSubject.doas
  - Authorization within doas loop
  - 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-Specific
Java Authentication and Authorization Service *(continued)*

- New in SDK 6:
  - Added new JAAS login module that enables users to perform authentication using credentials stored in an LDAP directory service
- z/OS documentation available at http://www-03.ibm.com/servers/eserver/zseries/software/java/jaas14.html

z/OS SAF Interfaces

- Java static class methods provide an interface to the z/OS Security Server using SAF (System Authorization 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()
- z/OS documentation available at http://www.ibm.com/servers/eserver/zseries/software/java/security14.html
- For information about additional RACF user and group administration APIs shipped with RACF, please see http://www-03.ibm.com/servers/eserver/zseries/zos/racf/racfjsec.html

IBM Java Generic Security Services - JGSS

- Used for secure message exchange between communicating applications
- Uniform access to security services atop a variety of underlying security mechanisms
  - Kerberos
  - Generic Security Service Application Program Interface (GSS-API) defined in RFC 2853
- Java GSS-API and JSSE2 provide same basic set of security features.... key differences:
  - GSS-API token-based application-driven communication and can use any communication channel; JSSE2 socket-based only

IBM Java Generic Security Services - JGSS *(continued)*

- Java GSS-API and JSSE2 provide same basic set of security features.... key differences (continued):
  - GSS-API allows the client to delegate its credentials to the server when using Kerberos
  - GSS-API permits selective encryption of messages; JSSE2 is all or none proposition per handshake
  - Protocol capabilities differ
    - JSSE2 supports SSL, TLS, and kerberos
    - JGSS supports GSS-API and kerberos
    - Some client/server exchanges require one or the other
- Code is common with other IBM platforms at SDK 1.4 level
  - 100% pure Java Implementation
  - Allows for application portability

IBM Java Generic Security Services - JGSS *(continued)*

- New in SDK 5:
  - Now supports the use of the udp_preference_limit property in the Kerberos configuration file (krb5.ini)
  - Now supports IPv6 addresses in Kerberos tickets. Previously, only IPv4 addresses were supported
  - Now supports Ticket Granting Ticket (TGT) renewal
    - allows long-running services to renew their TGTs automatically without user interaction or services restart
- New in SDK 6:
  - Added CipherText Stealing mode
  - Simple and Protected Negotiation (SPNEGO) mechanism included
- z/OS documentation available at http://www-1.ibm.com/servers/eserver/zseries/software/java/jgss14.html

IBM Java Certification Path - IBMCERTPATH

- Provides classes for creating, building, and validating certification paths (also known as "certificate chains").
- When checking the certificate's CRL Distribution Points extension, recognizes both HTTP and LDAP URLs.
- To enable CRL Distribution Points extension checking, use the system property com.ibm.security.enableCRLDP

IBM Java Certification Path - IBMCERTPATH *(continued)*

- New in SDK 5:
  - Supports checking a certificate's revocation status based on On-Line Certificate Status Protocol (OCSP)
- New in SDK 6:
  - Enhanced CRL validation and CRL processing to more closely comply with the PKIX Certificate and CRL Profile (RFC 3280) Section 6.3, entitled "CRL Validation"
  - Added com.ibm.security.enableDELTACRL system property to use both delta CRLs and complete CRLs if revocation checking is enabled by caller
  - Added com.ibm.security.enableAIAEXT system property to use LDAP URIs found in any Authority Information Access extensions within certificates on the certificate path
- z/OS documentation available at http://www-03.ibm.com/servers/eserver/zseries/software/java/certpath14.html

IBM Java Simple Authentication & Security Layer - IBMSASL

- New provider introduced in SDK 5
- A method for adding authentication support to connection-based protocols
- Defined to be mechanism-neutral: the application need not be hardwired into using any particular SASL mechanism.
- API supports both client and server applications.
- Allows applications to select the mechanism to use based on desired security features
- Also allows developers to use their own, custom SASL mechanisms

IBM Java Simple Authentication & Security Layer - IBMSASL *(continued)*

- Supports the following client and server mechanisms:
  - Client mechanisms
    - PLAIN (RFC 2595) - supports cleartext username/password authentication.
    - CRAM-MD5 (RFC 2195) - supports a hashed username/password authentication scheme.
    - DIGEST-MD5 (RFC 2831) - defines how HTTP Digest Authentication can be used as a SASL mechanism
    - GSSAPI (RFC 2222) - uses the GSSAPI for obtaining authentication information; supports Kerberos v5 authentication
    - EXTERNAL (RFC 2222) - obtains authentication information from an external channel (such as TLS or IPsec)
  - Server mechanisms
    - CRAM-MD5
    - DIGEST-MD5
    - GSSAPI (Kerberos v5)
- Documentation available at http://www-128.ibm.com/developerworks/java/jdk/security/60/secguides/saslDocs/ibm.sasl.provider.guide.html

IBM Java XML Digital Signature

- New provider in SDK 6 implementing JSR105
- W3C XML digital signature syntax and processing (http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/Overview.html)
- Standard set of APIs for XML digital signature services
  - Perform detached(over data external to sig element), enveloped(over data in same XML doc), and enveloping(over content within element of sig itself) signatures
  - Sign arbitrary binary data and include within an XML document
    - Yields XML Signature element containing or referencing signature data
- Supported algorithms:
  - Signature Algorithms: SHA1withDSA, SHA1withRSA
  - Macs: HmacSHA1
- Documentation available at http://www.ibm.com/developerworks/java/jdk/security/index.html

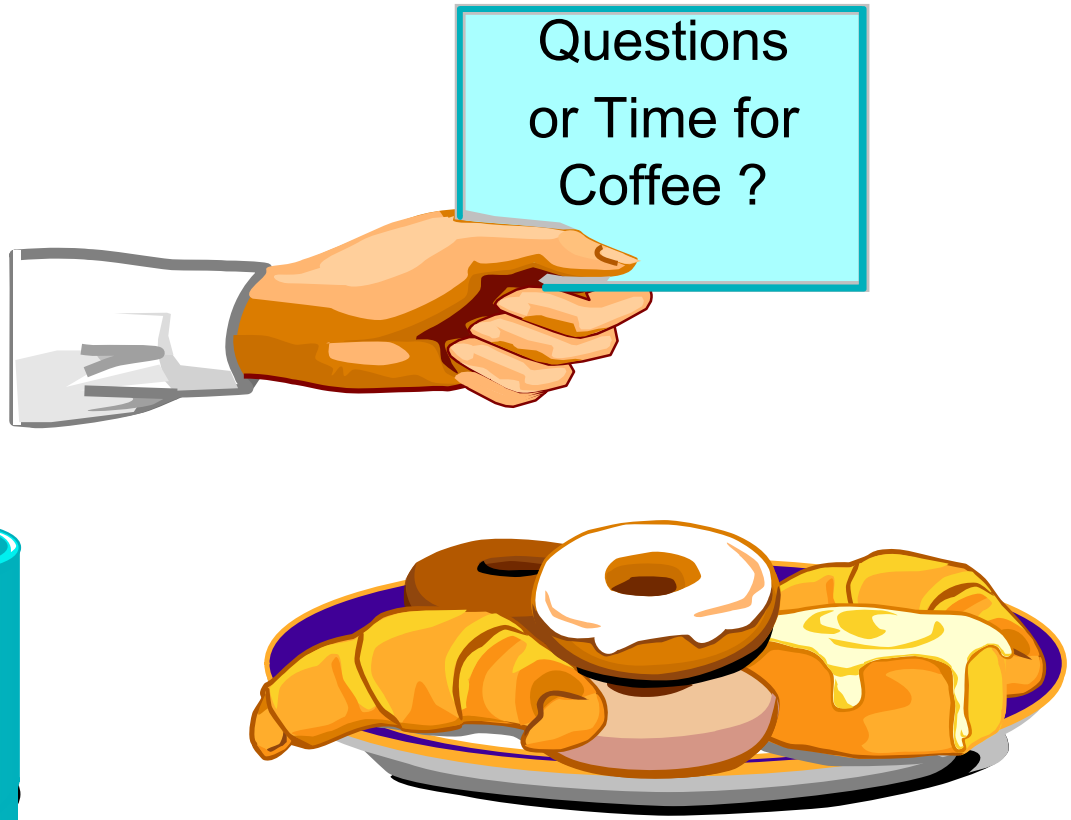
IBM Java XML Digital Encryption

- New provider in SDK 6 implementing JSR106
- W3C XML digital encryption syntax and processing (http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/Overview.html)
- Standard set of APIs for XML digital encryption services
  - Perform fine-grained, element-based encryption of fragments within an XML document
  - Encrypt arbitrary binary data and include within an XML document
    - Yields XML Encryption element containing or referencing the cipher data
- Supported algorithms:
  - Data encryption -Triple DES, AES
  - Key transport - RSA-v1.5, RSA-OAEP
  - Symmetric key wrap - Triple DES, AES
  - Transform – Base64, XPath, XSLT
- Documentation available at http://www.ibm.com/developerworks/java/jdk/security/index.html

Java Security components - Summary

- IBM Java SDK
  - IBMJCE - Java Cryptographic Extension
  - IBMJCECCA - Java Cryptographic Extension using CCA hardware cryptographic devices (z/OS only)
  - IBMJSSE2 - Java Secure Sockets Extension (SSL and TLS)
  - IBMPKCS11Impl - Public Key Cryptographic Standard #11
  - IBMJAAS - Java Authentication and Authorization Service
  - IBMJGSS - Generic Security Services - Kerberos, GSS-API
  - IBMCERTPATH - Certificate Path Verification
  - IBMSASL - Simple Authentication & Security Layer
  - IBMXMLCryptoProvider - XML Digital Signatures
  - IBMXMLEncProvider - XML Digital Encryption
  - SAF Interfaces (z/OS only)
- Provides Java Applications easy access to complex Security capabilities within the Java Platform framework (Java SE)
- Documentation: http://www.ibm.com/developerworks/java/jdk/security/index.html

Questions ?



Java Security - Appendix

- Miscellaneous Items of Interest.....

z/OS PKCS#11 Basics and Overview

- What is PKCS#11?
  - RSA's Public Key Cryptographic Standard #11
    - Smart card interface standard
  - APIs define cryptographic objects (keys, certificates, etc) and all the functions needed to use, create/generate, modify and delete those objects
- PKCS#11 Basic Terms
  - Token – Logical view of a crypto device, e.g., smart card (virtual or real)
  - Slot – Logical view of a card reader, numbered 0 - n
  - Slot number has 1 to 1 relationship with a token name
  - Session – Logical connection between application and token
  - Object – Item stored on token, e.g., certificate, key, etc
  - May be temporary (session) or persistent (token)
  - Attribute – Characteristic of an object, e.g., key label
  - PIN – Controls access to the entire token
  - User – Owns the data on a token by knowing the PIN
  - Security Officer (SO) – Person who initializes a token and assigns the user PIN (knows the SO PIN)

z/OS PKCS#11 Basics and Overview *(continued)*

- z/OS Implementation of PKCS#11
  - Implemented in ICSF on z/OS
  - Introduced in z/OS V1R9
  - Support includes:
    - Token data set (TKDS) - repository for keys and certs used by PKCS#11 applications
    - C APIs - supports subset of V2.20 PKCS#11 spec
    - Token management callable services (used by C APIs)
  - Access to tokens controlled by SAF profiles
    - New CRYPTOZ class in RACF
    - No personal identification numbers (PINs) are used
  - Token management accomplished using
    - RACF's RACDCERT
    - System SSL's gskkyman
    - ICSF ISPF PKCS#11 panels

z/OS Java Cryptography Extension - IBMJCECCA Hardware Exploitation

- Cryptographic hardware exploited by IBMJCECCA:
  - Cryptographic Co-processor Facility (CCF)
    - IBM CMOS G5, G6 and zSeries servers, **except** the zSeries z990 & System z9
    - DES, TDES, RSA, and various finance-industry-specific cryptographic operations
  - CP Assist for Cryptographic Function (CPACF)
    - IBM eServer zSeries z990 and System z9 servers
    - DES, TDES, MAC, and SHA-1 cryptographic operations

z/OS Java Cryptography Extension - IBMJCECCA Hardware Exploitation *(continued)*

- Cryptographic hardware exploited by IBMJCECCA (continued)
  - 4758 PCI Cryptographic Coprocessor (PCICC) card available on CMOS G5, G6 and zSeries servers, except zSeries z990 & System z9
    - financial processes (i.e. PIN processing) and key management operations
    - DES, RSA, and DSA cryptographic operations
    - System (master) Key and retained key storage
  - IBM e-business PCI Cryptographic Accelerator (PCICA) card available on zSeries servers
    - Usable only for SSL crypto function (decryption of pre-master secret from under server's public key)

z/OS Java Cryptography Extension - IBMJCECCA Hardware Exploitation *(continued)*

- Cryptographic hardware exploited by IBMJCECCA (continued)
  - IBM PCI-X Cryptographic Coprocessor (PCIXCC) card available on zSeries z990 & System z9 servers
    - replacement for both the PCICC and the CCF
    - DES, TDES, RSA, SHA-1 cryptographic operations, modular-exponentiation for RSA, DSA
  - IBM Cryptographic Express2 (CEX2C Coprocessor/CEX2A Accelerator) Feature available on zSeries z990 & System z9 servers
    - replacement for PCIXCC and PCICA
    - equivalent functions with equal or greater performance
- More information at
  - http://www.ibm.com/security/cryptocards/
  - http://www-03.ibm.com/systems/z/security/pdf/Web_z9_Crypto_Rel_01202006_X.pdf

Java Security - Glossary

- Advanced Encryption Standard - an improved block cipher successor to DES
- AES - Advanced Encryption Standard
- Asymmetric Key Cipher - Also known as public-private key cryptography system; a cryptography system that uses two different keys to lock and unlock (encrypt and decrypt) messages and files. See Public-private Key Cryptography System.
- Authentication - the process of verifying that a file or message has not been altered in route from the distributor to the recipient(s).
- Block Cipher - A method for encrypting data in chunks (several or many contiguous bits) as opposed to encoding bit-by-bit like a stream cipher.
- Blowfish - A block cipher that employs the asymmetric key model.
- CBC - Cipher Block Chaining: A method of operating a symmetric block cipher that uses feedback to combine previously generated ciphertext with new plaintext.
- CCA - Common Cryptographic Architecture
- Certificate- A certificate is a data file that identifies an individual, organization, or business. Certificates are obtained from specialized certificate-issuing companies such as VeriSign, and can be used to encrypt data and/or confirm the certificate owner's identity.
- Certificate Revocation List - When using public key infrastructure, this list enumerates revoked certificates for subscribers, along with the reason(s) for revocation

Java Security - Glossary *(continued)*

- CFB - Cipher Feedback: A block cipher mode that processes small increments of plaintext into ciphertext, instead of processing an entire block at a time.
- Cipher - a cryptographic algorithm used to encrypt and decrypt files and messages.
- Ciphertext Stealing mode - a technique used in block cipher mode operations for encrypting messages which aren't evenly divisible into blocks, without yielding any ciphertext expansion.
- Common Cryptographic Architecture - a collection of software components that provide common application interfaces to secure, high-speed cryptographic services on various platforms via hardware cryptographic devices
- CMVP - Cryptographic Module Validation Program
- CRL - Certificate Revocation List
- Cryptographic Module Validation Program - The Computer Security Division at NIST maintains a number of cryptographic standards, and coordinates validation programs for many of those standards. The Cryptographic Module Validation Program (CMVP) encompasses validation testing for cryptographic modules and algorithms.
- DES - Data Encryption Standard. A cipher developed by the United States government in the 1970s to be the official encryption algorithm of the U.S. (FIPS 46-3 proposed to be withdrawn by National Institute of Standards and Technology http://csrc.nist.gov/Federal-register/July26-2004-FR-DES-Notice.pdf.)
- Digest Authentication - Per IETF RFC # 2617, an authentication mechanism which verifies that both parties to a communication know a shared secret (a password) using a scheme based on cryptographic hashes.

Java Security - Glossary *(continued)*

- Digital Signature - A small piece of code that is used to authenticate the sender of data. Digital signatures are created with encryption software for verification purposes. A private key is used to create a digital signature, and a corresponding public key can be used to verify that the signature was really generated by the holder of the private key.
- ECB - Electronic Codebook Encryption: A block cipher mode (each block is encrypted individually) that uses no feedback.
- ElGamal - An asymmetric cipher algorithm.
- Federal Information Processing Standards - Under the Information Technology Management Reform Act (Public Law 104-106), the Secretary of Commerce approves standards and guidelines that are developed by the National Institute of Standards and Technology (NIST) for Federal computer systems. These standards and guidelines are issued by NIST as Federal Information Processing Standards (FIPS) for use government-wide.
- FIPS - Federal Information Processing Standards
- Generic Security Service Application Program Interface - Per IETF RFC 2078, provides security services to callers in a generic fashion, supportable with a range of underlying mechanisms and technologies and hence allowing source-level portability of applications to different environments.
- GSS-API - Generic Security Service Application Program Interface
- HMAC - Keyed-Hash Message Authentication Code
- ICSF - Integrated Cryptographic Services Facility

Java Security - Glossary *(continued)*

- Integrated Cryptographic Services Facility - Asoftware component of z/OS which provides the administrative interface and a large set of application interfaces to cryptographic hardware and services.
- IPSec - IP Security
- IP Security - a set of protocols developed by the IETF to support secure exchange of packets at the IP layer
- Kerberos - A protocol which is used to enable users to perform secure logins to a network of inter-communicating computers
- Key - a collection of bits, usually stored in a file, which is used to encrypt or decrypt a message.
- Key Agreement - A key exchange mechanism used by two parties to agree on a secret session key.
- Keyed-Hash Message Authentication Code - A FIPS standard specifying the use of cryptographic hash functions in the algorithms which generate a message authentication code.
- LDAP - Lightweight Directory Access Protocol
- MAC - Message Authentication Code
- MD5 - Message Digest 5
- Message Authentication Code - a message digest which uses a key to create the message digest value, useful for ensuring the integrity of data being sent over an insecure network

Java Security - Glossary *(continued)*

- Message Digest - A number generated by applying a mathematical algorithm to any arbitrary data, used to verify data integrity
- OAEP - Optimal Asymmetric Encryption Padding
- OCSP - Online Certificate Status Protocol
- OFB - Output Feedback: A block cipher mode that uses feedback similar to the Cipher Feedback (CFB) mode, differing in how shift register is filled.
- Online Certificate Status Protocol - Per IETF RFC # 2560, a protocol which enables applications to determine the (revocation) state of an identified certificate, in lieu of using a CRL.
- Optimal Asymmetric Encryption Padding - An encryption padding scheme used to process plaintext before asymmetric encryption (i.e. RSA)
- PBE - Password-Based Encryption (PKCS#5) - a cipher which encrypts using a secret key derived from a password
- PCBC - Propagating Cipher Block Chaining: much like CBC mode, except that both the plaintext and the ciphertext of the previous block are used, rather than just the ciphertext.
- PKDS - Public-private Key Data Set
- PKIX - An IETF working group established to develop new standards apropos to the use of X.509-based Public Key Infrastructures in the Internet.

Java Security - Glossary *(continued)*

- Private Key - the secret key of a public-private key cryptography system. This key is used to "sign" outgoing messages, and is used to decrypt incoming messages.
- Public Key - the public key of a public-private key cryptography system. This key is used to confirm "signatures" on incoming messages or to encrypt a file or message so that only the holder of the private key can decrypt the file or message.
- Public-private Key Cryptography System - a cryptography system that uses two different keys to lock and unlock (encrypt and decrypt) messages and files. The two keys are mathematically linked together. An individual's public key is distributed to other users and is used to encrypt messages to the individual. The individual keeps the private key secret and uses it to decrypt messages sent with the public key.
- Public-private Key Data Set - An ICSF-managed dataset used to store asymmetric cryptographic keys
- RC2 - Rivest Cipher #2 - A block cipher algorithm owned by RSA Data Security, Inc.
- RC4 - Rivest Cipher #4 - A stream cipher algorithm owned by RSA Data Security, Inc.
- RSA - Rivest-Shamir-Adelman: a family of algorithms that employ the asymmetric key model.
- SAF - System Authorization Facility
- SASL - Simple Authentication & Security Layer
- Secure Hash Algorithm - An algorithm developed by NIST to generate message digests.

Java Security - Glossary *(continued)*

- Secure Sockets Layer - A protocol for encrypting information before being transmitted over the Internet.
- SHA1 - Secure Hash Algorithm
- Simple Authentication & Security Layer - Per IETF RFC # 2222, a method for adding authentication support to connection-based protocols.
- Stream Cipher - A method of encrypting data bit-by-bit, as opposed to encoding a contiguous chunk of data all at once like a block cipher.
- SSL - Secure Sockets Layer
- Symmetric Key - the key that is used to encrypt a file or message is the same key that is used to decrypt the file or message.
- System Authorization Facility - A part of the z/OS operating system which provides standard interfaces to services of external security managers for purposes of performing access control checking and authentication

Java Security - Glossary *(continued)*

- TKDS - Token Data Set
- TLS - Transport Layer Security
- Transport Layer Security - Per IETF RFC # 2246, a protocol which provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.
- TripleDES - A method of improving the strength of the DES algorithm by using it three times in sequence with different keys. Also known as DESede.
- Trust Manager - A service for managing cryptographic certificates of trust
- URI - Uniform Resource Identifier
- XML - Extensible Markup Language
- XPath - XML Path Language
- XSLT - Extensible Stylesheet Language Transformations