z/TPF V1.1

TPF Users Group Fall 2008

z/TPF Public Key Infrastructure Support

Name: Mark Gambino
Venue: Communications Subcommittee

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

© 2008 IBM Corporation

Agenda

- **Brief review of symmetric key cryptography and current z/TPF secure key management support**
- **Public key cryptography education and use cases**
- **OpenSSL and public key cryptography**
- **z/TPF PKI support preview**

© 2008 IBM Corporation

Symmetric Key Cryptography

- **The same key that encrypts the data is also used to decrypt the data**
- **Algorithms include variations of**
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)
- **Can be used to encrypt small or large amounts of data**

© 2008 IBM Corporation

Current z/TPF Secure Key Management Support

- **Ability to create, manage, and use symmetric keys in a secure manner**
- **Supports DES, Triple-DES, AES-128, AES-256**
- **Uses hardware acceleration to perform cryptographic operations**
  - Scales to tens or hundreds of thousands of secure key operations per second on a single z/TPF image

© 2008 IBM Corporation

Agenda

- **Brief review of symmetric key cryptography and current z/TPF secure key management support**
- **Public key cryptography education and use cases**
- **OpenSSL and public key cryptography**
- **z/TPF PKI support preview**

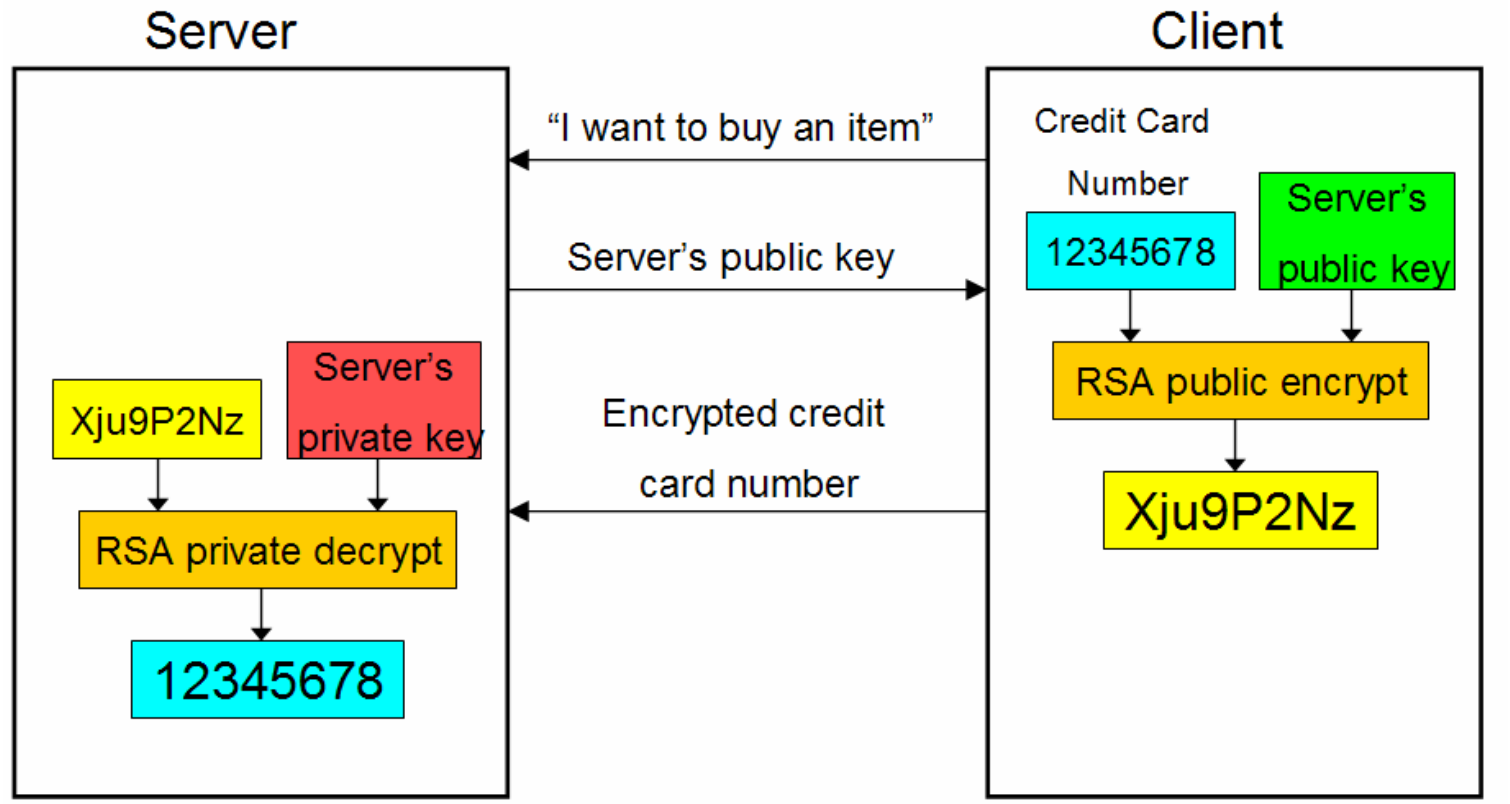
© 2008 IBM Corporation

Public Key Cryptography

- **Also known as asymmetric key cryptography**
- **Public key pair**
  - Public key
    - Can be openly distributed
  - Private key
    - Must be kept secret/protected
- **Any data encrypted with a public key can only be decrypted using the corresponding private key**
- **Any data encrypted with a private key can only be decrypted using the corresponding public key**
- **Commonly used algorithm is Rivest, Shamir, and Adleman (RSA)**

© 2008 IBM Corporation

Simple Public Key Cryptography Example



© 2008 IBM Corporation

Simple Public Key Cryptography Example Steps

1. **Client wants to purchase an item from the server**
2. **Server sends its public key to the client**
3. **Client encrypts its credit card number using the server's public key**
4. **Client sends the encrypted credit card number to the server**
5. **Server decrypts the client's credit card number using the server's private key**

© 2008 IBM Corporation

Limitations of Public Key Cryptography

- **Can only encrypt small amounts of data**
  - For example, a 1024-bit RSA public key can only encrypt up to 128 bytes of data
- **Is much more computationally expensive compared to symmetric key cryptography**
  - True even when using hardware cryptography. For example on an IBM System z10 using a fixed data size:
    - One Crypto Express2 accelerator can perform up to 3000 private key decrypt operations per second (RSA 1024-bit)
    - One Central Processor Assist for Cryptographic Functions (CPACF) coprocessor can perform over 600,000 symmetric key operations per second (AES-256)

© 2008 IBM Corporation

Best Practice for Encrypting Large Amounts of Data

- **To encrypt large amounts of data efficiently:**
  1. Client creates a symmetric key (KEY1)
  2. Client encrypts KEY1 using the server's public key and sends the encrypted KEY1 to the server
  3. Server decrypts KEY1 using its private key
  4. Client and server exchange data encrypted with KEY1
- **Secure Sockets Layer (SSL) works this way**
- **Another variation is for an enterprise key manager (EKM) to create the symmetric key and distribute it to both the client and server nodes using public key cryptography**
- **Public key cryptography solves the problem of symmetric key distribution/exchange over unsecure (public) networks**

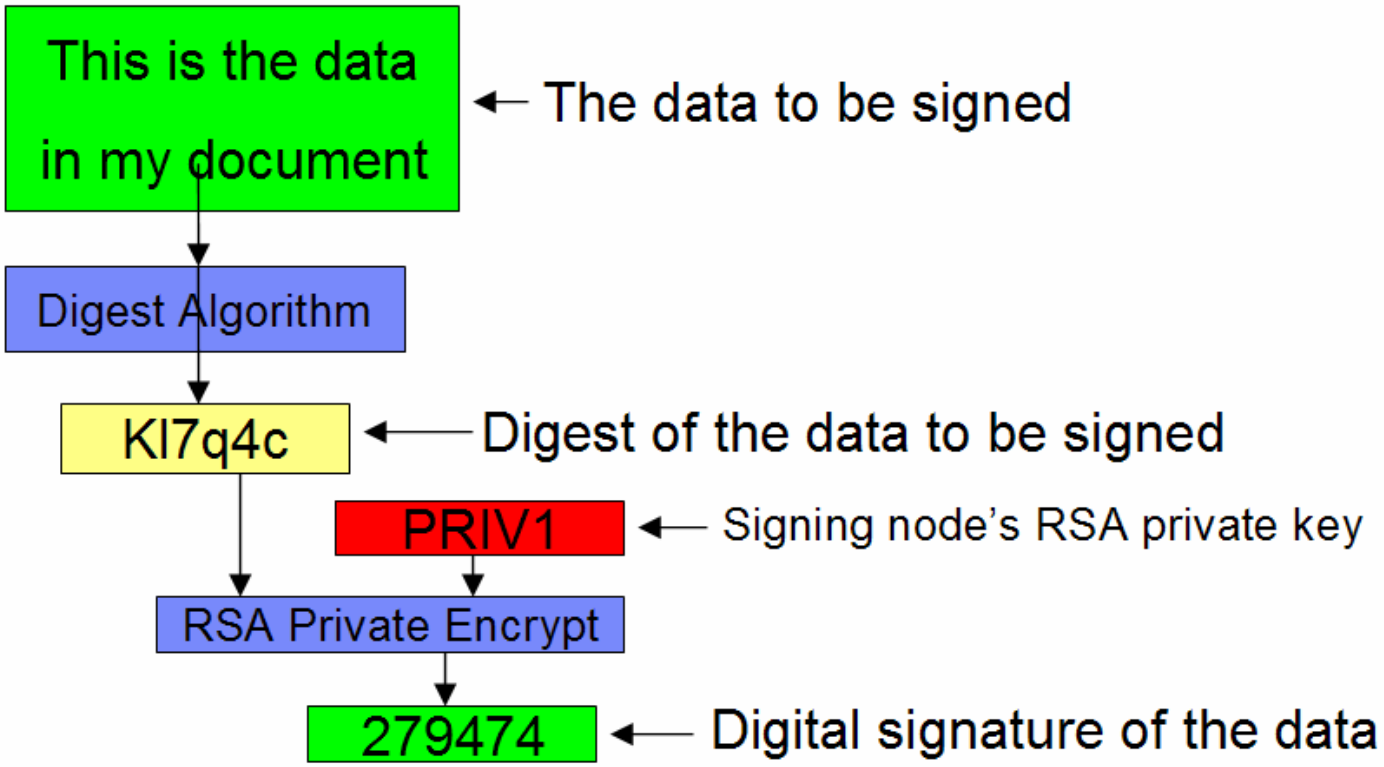
© 2008 IBM Corporation

Digital Signatures

- **Another important use case for public key cryptography is digital signatures**
- **Ability for the receiver of a message (data) to verify that the message was sent by the who the sender claims to be and that the message has not been altered after the sender built and signed the message (non-repudiation)**
- **The signed data can be in clear, encrypted, or a combination of both**

© 2008 IBM Corporation

Creating a Digital Signature



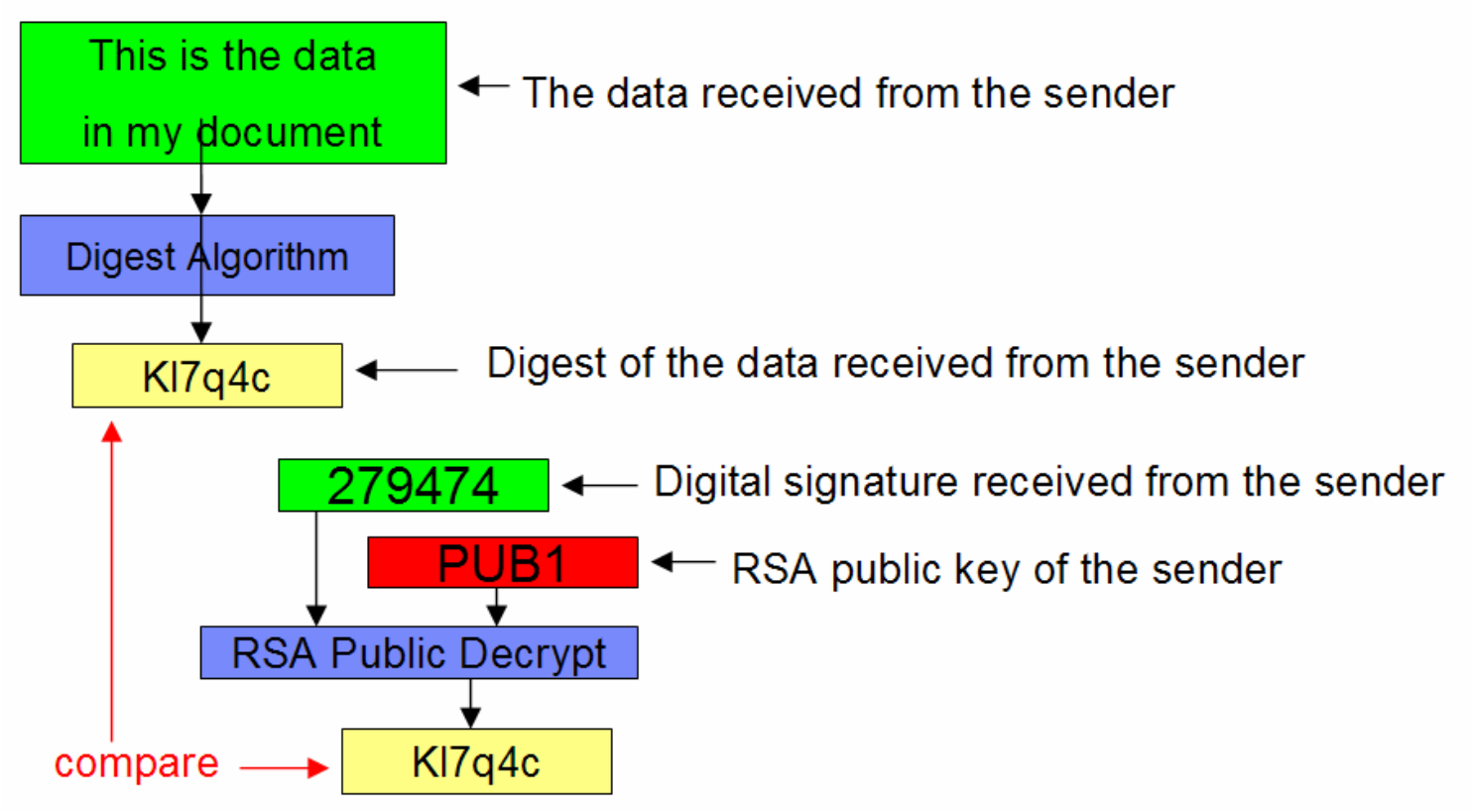
© 2008 IBM Corporation

Steps for Creating a Digital Signature

1. **Create a digest of the data to be signed**
   - Produces a fixed-length digest value
     - 20 bytes for SHA-1, 32 bytes for SHA-256
2. **Encrypt the digest using your RSA private key and the result is a digital signature of the data**
3. **Attach/append the digital signature to the data**
4. **Send the data and its digital signature to the remote partner**

© 2008 IBM Corporation

Verifying a Digital Signature



© 2008 IBM Corporation

Steps for Verifying a Digital Signature

1. **Create a digest of the data received from the sender**
2. **Take the digital signature received from the sender and decrypt it using the sender's RSA public key.**
3. **Compare the output of steps 1 and 2. If the values match, the digital signature is valid, meaning the sender did indeed send this data and the data has not been altered after it was signed by the sender**

© 2008 IBM Corporation

Considerations for Distributing Public Keys

- **It is not safe to just send your RSA public key by itself over unsecure networks to remote partners**
  - "Man in the middle" attacks can replace the real public key with the attacker's public key causing the receiver to encrypt sensitive data that the attacker will be able to decrypt/access
- **The answer to public key distribution is by using digital certificates**
  - Send your digital certificate to remote partners
    - Digital certificate contains your public key

© 2008 IBM Corporation

Digital Certificates

- **Uniquely identify a person, node, or application**
- **Are created/signed by a trusted authority called a certificate authority (CA)**
- **Have expiration date**
- **Analogous to passports issued by governments to its citizens**

© 2008 IBM Corporation

X509 version 3 Digital Certificate Contents

- **Data Section**
  - Serial number
  - Subject information (who the certificate identifies)
  - Public key of the subject
  - Issuer information (what CA created this certificate)
  - Validity period (start and end dates that the certificate can be used)
  - Optional certificate extensions to provide more information about the subject
- **Signature Section**
  - Digital signature of the information in the data section
  - Signed using the CA's private key

© 2008 IBM Corporation

Public Key Infrastructure (PKI)

- **Enables users of an unsecure (public) network to securely and privately exchange data**
- **Uses public key cryptography**
- **Public keys are shared/distributed using digital certificates that are created/signed by a trusted authority called a certificate authority (CA)**

© 2008 IBM Corporation

Agenda

- **Brief review of symmetric key cryptography and current z/TPF secure key management support**
- **Public key cryptography education and use cases**
- **OpenSSL and public key cryptography**
- **z/TPF PKI support preview**

© 2008 IBM Corporation

OpenSSL Programming Model

- **Server application issues APIs specifying the file names (in the file system) that contain:**
  - The server's private key
  - The server's digital certificate
- **If client authentication is being used, the client application must also issue APIs specifying the file names that contain:**
  - The client's private key
  - The client's digital certificate

© 2008 IBM Corporation

Private Key Files in OpenSSL

- **Having the private key in the clear in the file system or flowing across the network is a security risk**
- **When the private key file is created, you can optionally encrypt the contents (the private key) with a user specified password**
  - Applications must issue an OpenSSL API specifying the password that is needed to decrypt the information in the private key file

© 2008 IBM Corporation

Current SSL Setup Procedures for z/TPF

1. **Create an RSA key pair on another platform**
   - This also creates the RSA private key file
2. **Create a certificate request on another platform using the RSA public key from step 1 as input**
3. **Send the certificate request to a CA that will create and sign the certificate**
4. **Send (FTP) the certificate file and private key file to the z/TPF file system**

© 2008 IBM Corporation

OpenSSL Private Key Security Risks

- **SSL protocol relies/requires that the value of private key not be known by anyone other than the systems code on the node to which this private key is assigned**
- **Private key file is in the clear when sent across the network to z/TPF or in the file system**
  - Encrypting the private key file contents with a password alleviates this… somewhat
    - Operator that created the private key file knows the password
    - Application programmers also know the password as it must be coded in the applications that use this private key file
- **When the application issues the API telling the system the name of the private key file, the private key is brought into memory**
  - Private key value is in the clear in the application program's memory space
- **If the private key is compromised, anyone who finds out the private key value can access any data flowing over SSL**

© 2008 IBM Corporation

Agenda

- **Brief review of symmetric key cryptography and current z/TPF secure key management support**
- **Public key cryptography education and use cases**
- **OpenSSL and public key cryptography**
- **z/TPF PKI support preview**

© 2008 IBM Corporation

z/TPF PKI Support Basics

- **Allows you to create, manage, and use RSA key pairs in a secure manner**
  - Extends z/TPF secure key management support
- **Supports 1024-bit and 2048-bit RSA key pairs**
- **Key pair is referenced by name**
- **Private key value is secured – not visible to operators, applications, coverage, and so on**
- **Public key value is available to anyone**

© 2008 IBM Corporation

Creating an RSA Key Pair

- **Use the ZPUBK GENERATE command**
  - Specify the name of the key pair (for example, KEYPAIR1) which is how subsequent operator commands and APIs will reference/use this key pair
  - Specify the key length (1024-bit or 2048-bit)
  - RSA key pair is created and added to the PKI keystore
- **Issue additional operator commands to backup the keystore and activate the key pair making it available for use**

© 2008 IBM Corporation

Creating a Certificate

1. **z/TPF operator creates an RSA key pair called KEYPAIR1**
2. **Create a file (called myinfo.cfg in this example) containing the subject information needed to create a certificate request**
3. **Issue the ZPUBK REQCERT command. Input includes:**
   - Key pair name (KEYPAIR1) that says which public key to use
   - Name of the file (myinfo.cfg) containing the subject information
   - Name of the file (mycert.fil) into which to build the certificate request (PKCS #10 format)
4. **Send (FTP) the certificate request to the CA who will create the digital certificate**
5. **From the CA, send (FTP) the certificate to your z/TPF system**

© 2008 IBM Corporation

Self-Signed Certificates

- **A self-signed certificate is where the subject and issuer of the certificate are the same**
- **In production typically the only nodes that have/create self-signed certificates are CAs**
- **It is often convenient to have self-signed certificates for test systems**
- **To create a self-signed certificate on z/TPF**
  - Specify the SSIGNED option on the ZPUBK REQCERT command
  - Creates a certificate (self-signed) rather than a certificate request

© 2008 IBM Corporation

Solving OpenSSL Private Key Concerns

- **OpenSSL programming model cannot be changed**
  - Would break lots of middleware built on top of OpenSSL
- **SSL applications will be able to use an RSA key pair generated by z/TPF**
- **Application program still uses standard OpenSSL APIs to indicate the name of the file that contains the private key**
  - If the file name path starts with a special prefix (/tpfpubk), this tells z/TPF that the name that follows is really the name of the RSA key pair to use and not to try and open/use a file in the file system

© 2008 IBM Corporation

Solving OpenSSL Private Key Concerns (continued)

- **z/TPF operator creates an RSA key pair called KEYPAIR1**
- **To use this RSA key pair, an SSL application on z/TPF issues the *SSL_CTX_use_PrivateKey_file* API with the private key file name set to:**
  - /tpfpubk/keypair1.pem
- **The private key value is not copied into the application program's memory space**
  - Only the key pair name (KEYPAIR1) is saved in the SSL structure in the application's memory space

© 2008 IBM Corporation

Middleware using OpenSSL that Accesses Private Key File

- **Middleware using OpenSSL typically has its own configuration file that includes:**
  - The file name of the RSA private key
  - The file name of certificate
- **Some middleware opens/reads the private key file to determine whether the private key is encrypted and if so, prompts the operator for the password**
- **Some middleware also compares the public key information in the private key file to the public key in the certificate to make sure they match**

© 2008 IBM Corporation

Solving the Middleware Access Problems

- **When you create an RSA key pair on z/TPF**
  - A dummy private key file is created in the /tpfpubk directory
  - This file includes:
    - An unencrypted private key (no password needed)
      - This is NOT the real private key value!
      - Part of fake private key value includes the key pair name
    - The real public key value
- **Allows middleware to continue to work without any changes to the middleware**
  - Middleware does not have access to the real private key value

© 2008 IBM Corporation

Data Encryption Outside the Scope of SSL

- **If you create a symmetric key on z/TPF and one z/TPF complex is the only one that needs to use that key**
  - Key distribution is not an issue
  - Example – z/TPF application encrypts data before writing it out to the z/TPF database and that application reads the data (and decrypts) it later on
- **If you want exchange data with another platform and the data is encrypted at the application level using a symmetric key**
  - A key distribution mechanism is needed

© 2008 IBM Corporation

Solving the Symmetric Key Distribution Problem

- **z/TPF will support RSA key wrapping to securely import symmetric keys**
- **Remote key manager that creates the symmetric key (KEY1) will wrap (encrypt) KEY1 using z/TPF's public key**
- **The remote key manager sends the encrypted KEY1 value to z/TPF**
  - How this is done is not architected (not standardized)
- **API on z/TPF will unwrap (decrypt) the KEY1 value (using z/TPF's private key) and add KEY1 to the symmetric key keystore**
  - The KEY1 value is never in the clear during the key exchange

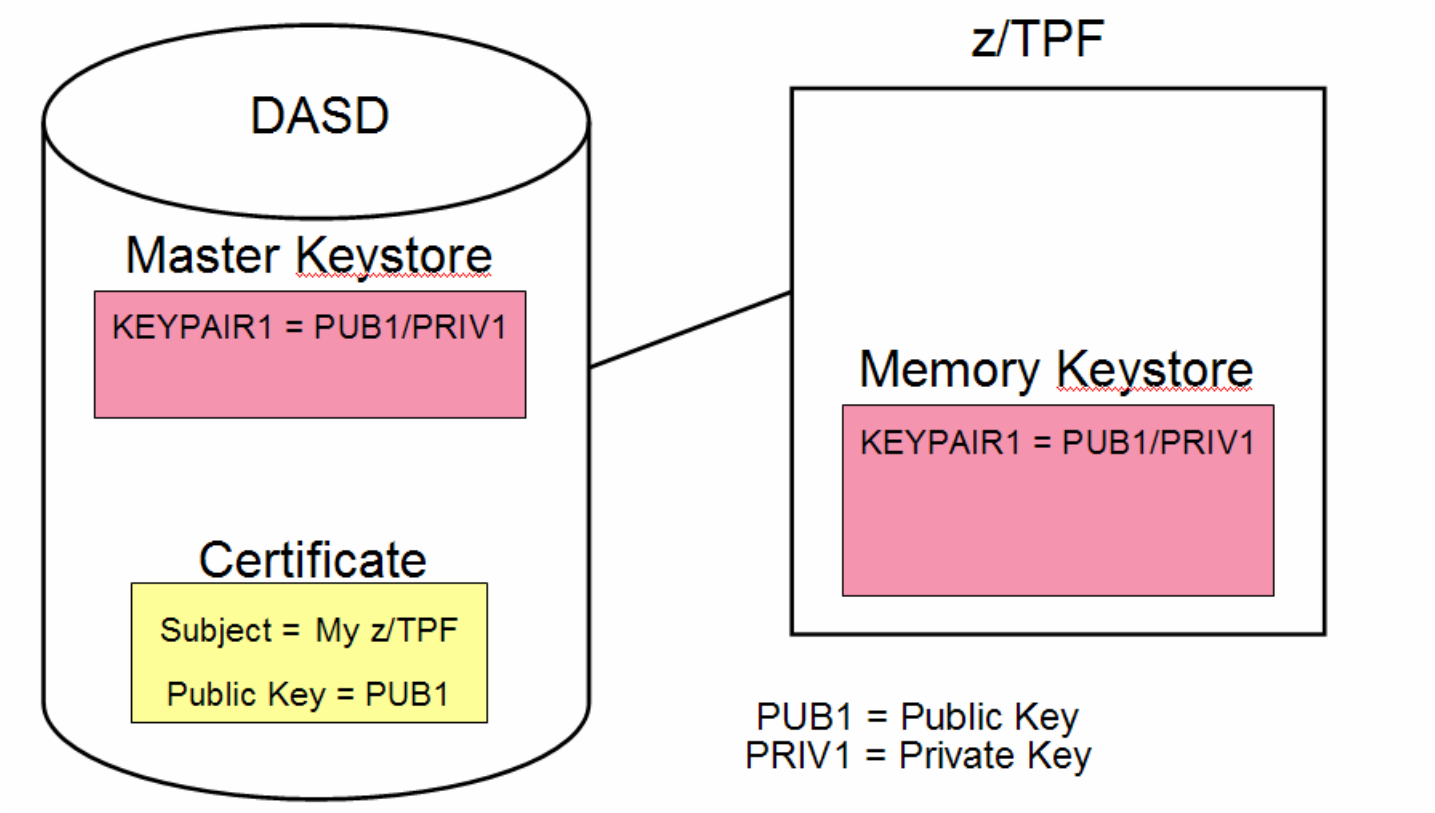
© 2008 IBM Corporation

How Does the Remote Key Manager get Public Key Values

- **Remote key manager needs z/TPF's public key to be able to securely send a wrapped symmetric key to z/TPF**
- **Can extract the public key on z/TPF**
  - Use the ZPUBK EXTRACT command
  - Creates a public key file containing the public key
  - Can send (secure FTP) the public key file to the remote key manager
- **The other (and more commonly used) option is to send z/TPF's certificate to the remote key manager**
  - Certificate contains z/TPF's public key

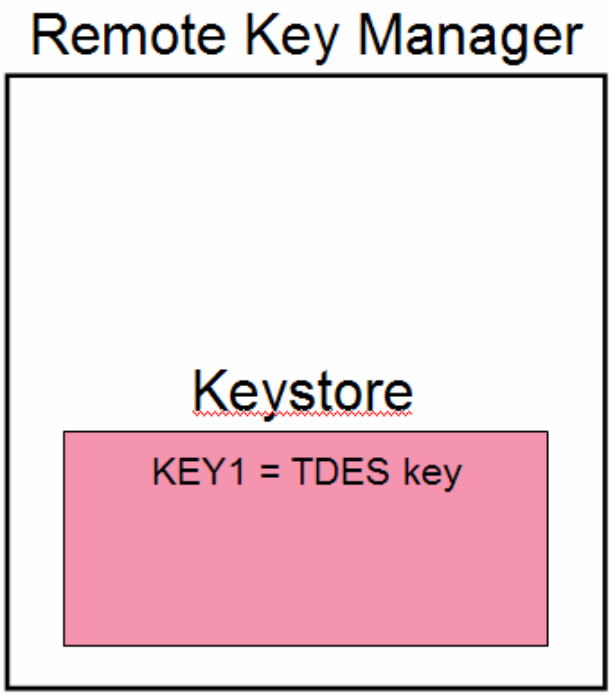
© 2008 IBM Corporation

Secure Key Import Example – Step 1
Create RSA Public Key Pair (KEYPAIR1) and Certificate



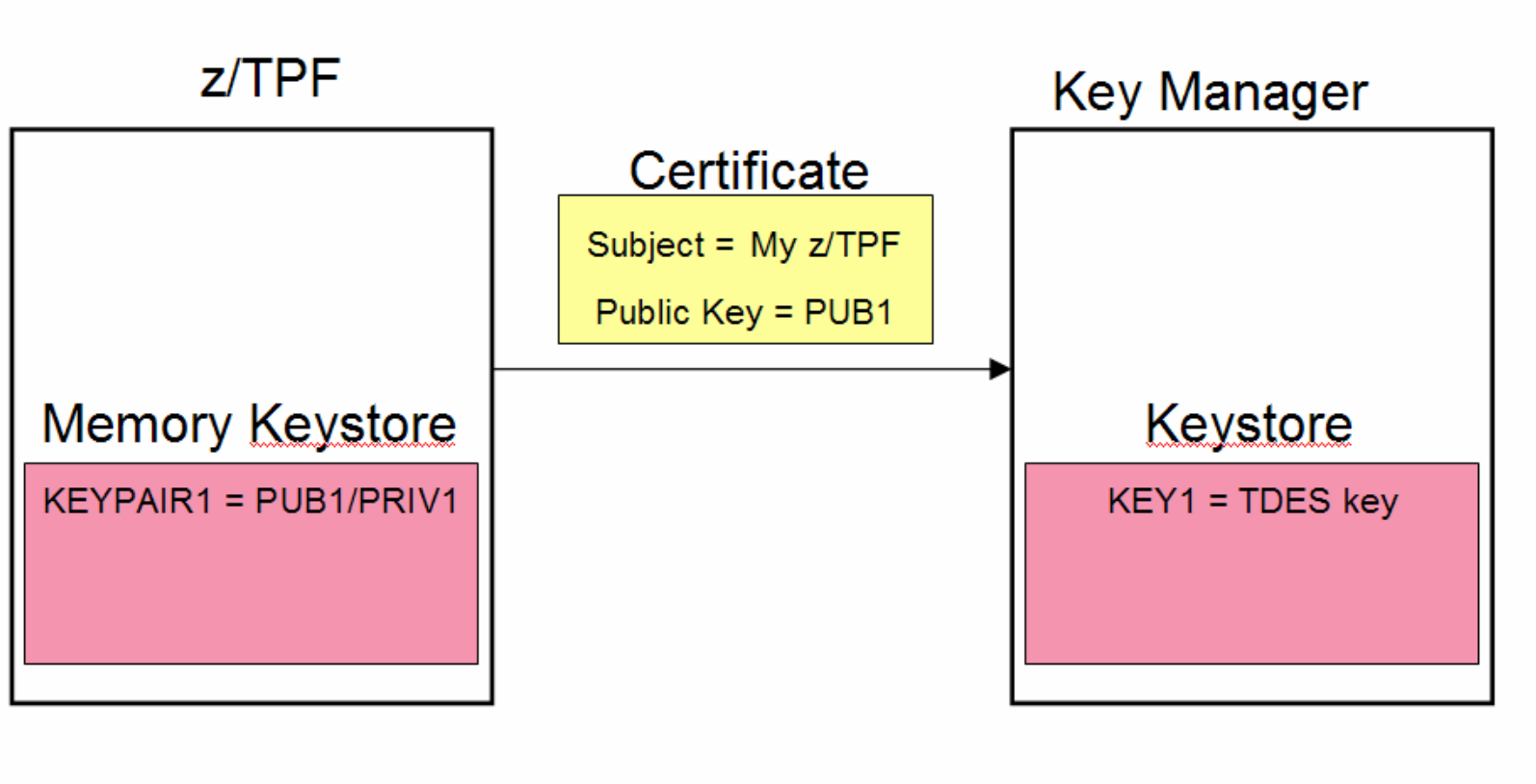
© 2008 IBM Corporation

Secure Key Import Example – Step 2
Key Manager Creates Symmetric Key (KEY1) Using TDES



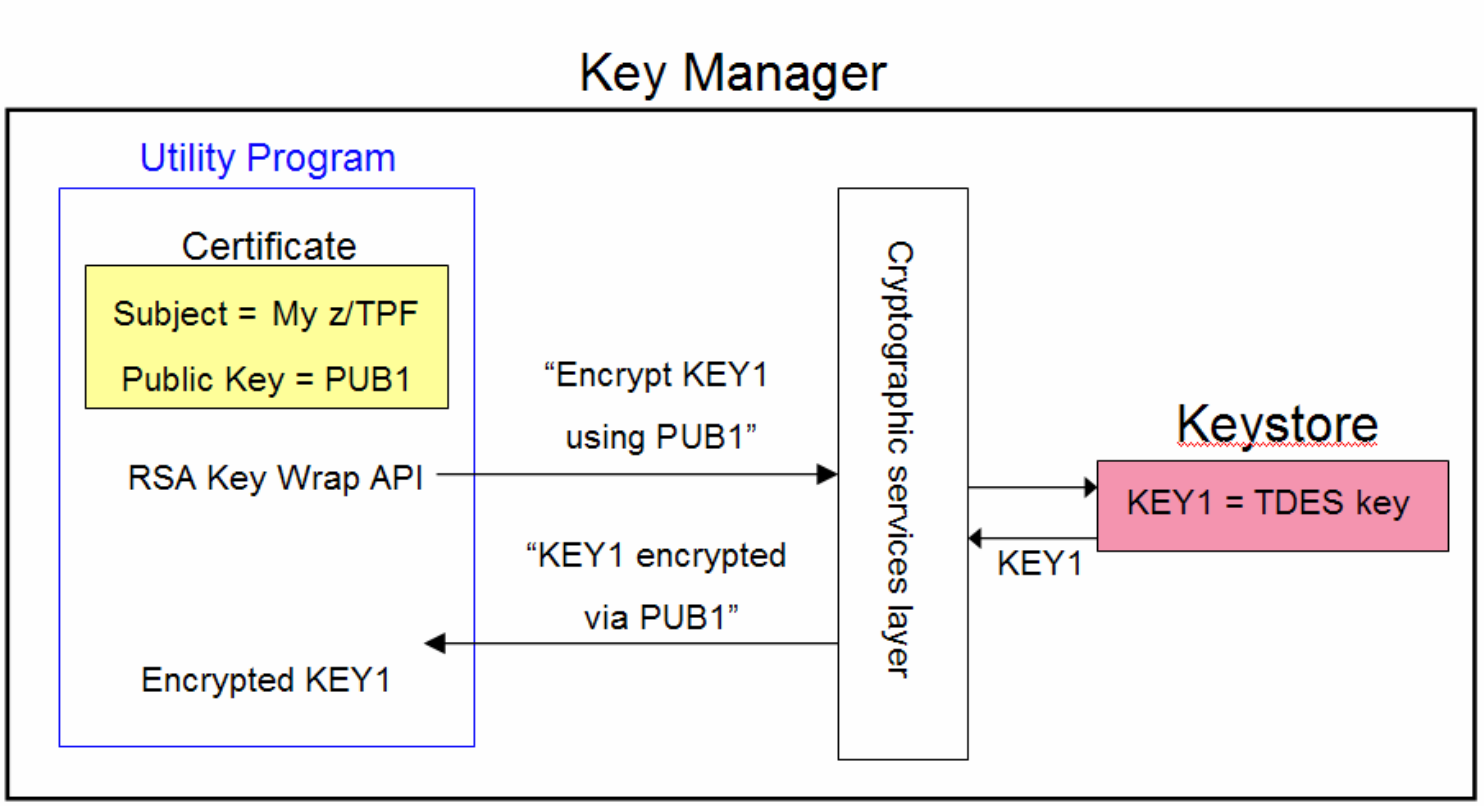
© 2008 IBM Corporation

Secure Key Import Example – Step 3
Send z/TPF's Public Key (Certificate) to Key Manager



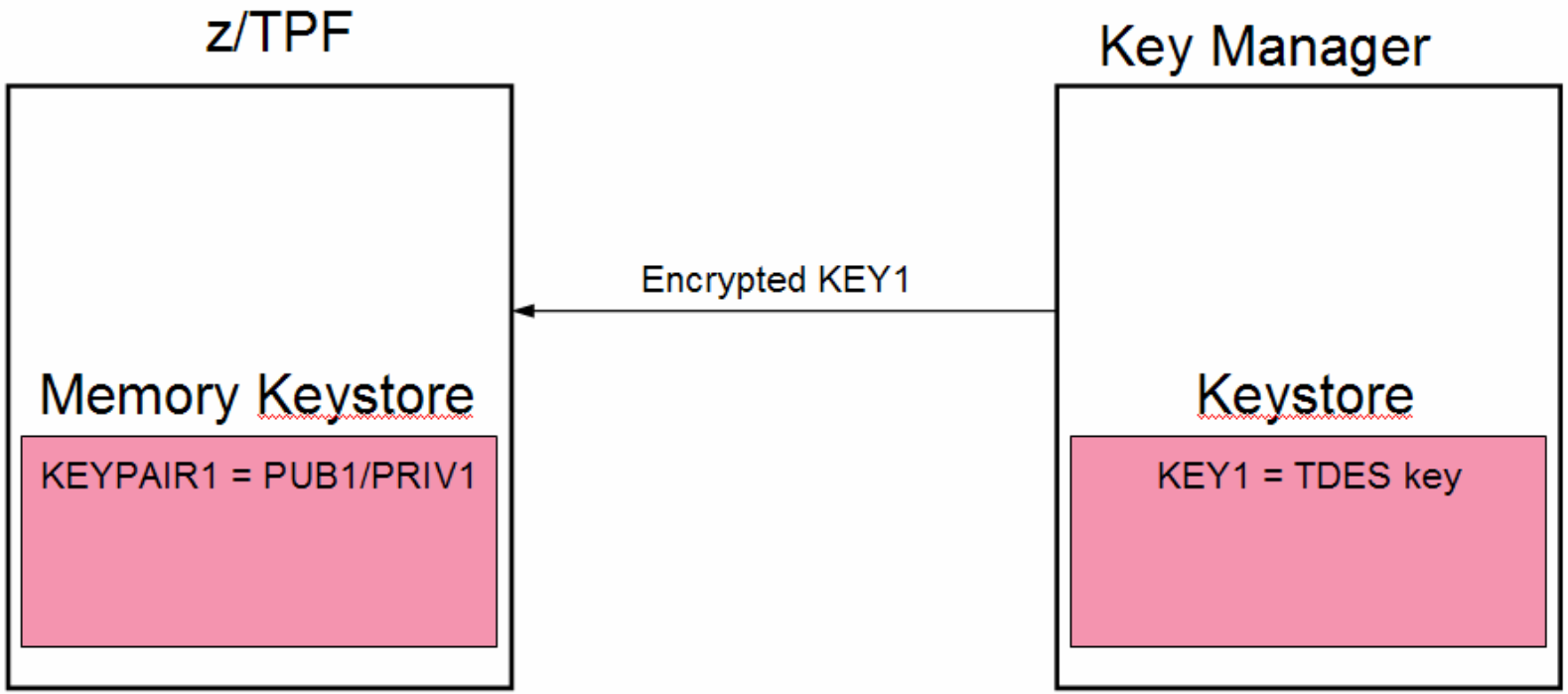
© 2008 IBM Corporation

Secure Key Import Example – Step 4
Wrap Symmetric Key (KEY1) Using z/TPF's Public Key



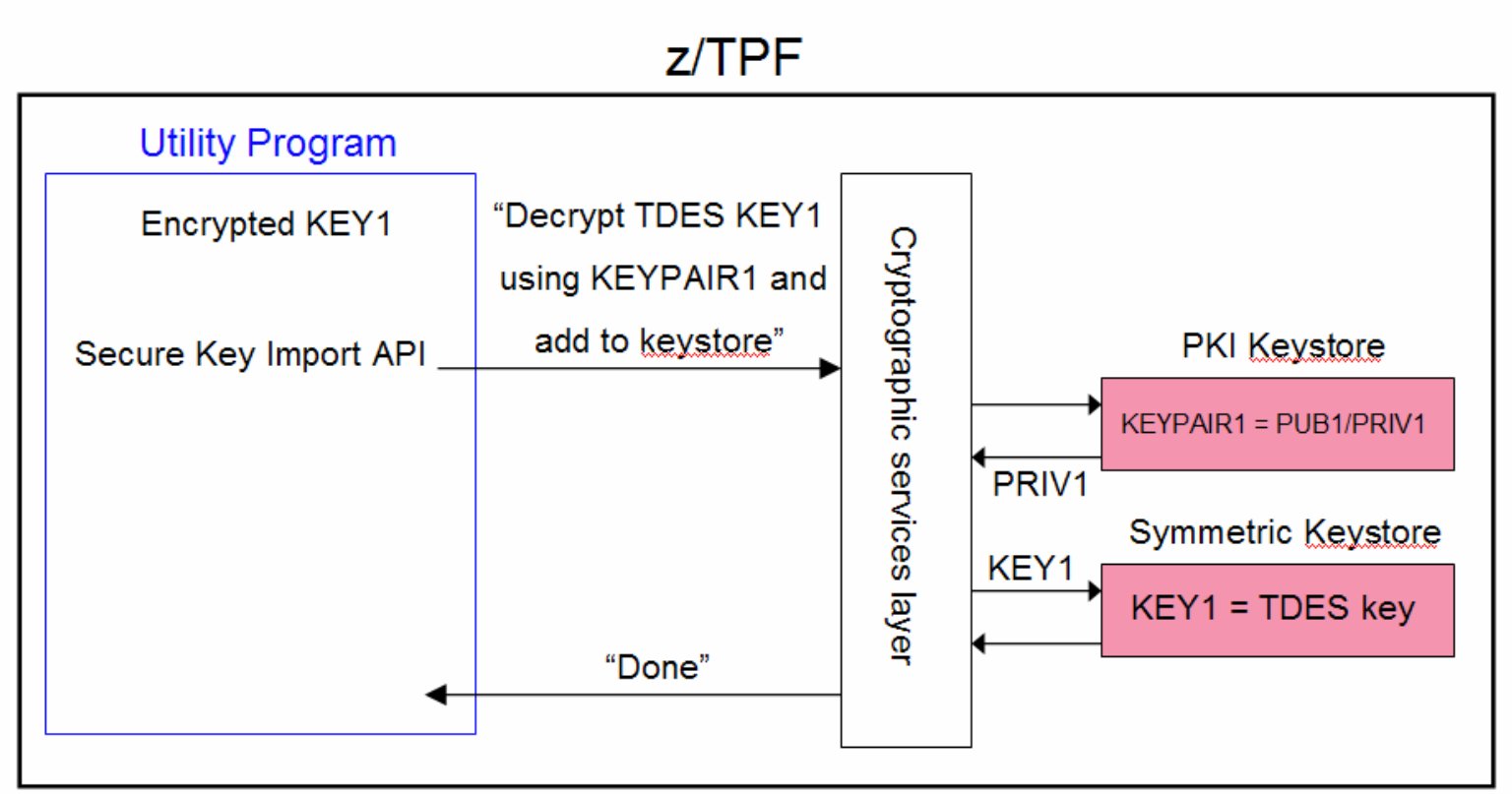
© 2008 IBM Corporation

Secure Key Import Example – Step 5
Send Wrapped Symmetric Key to z/TPF



© 2008 IBM Corporation

Secure Key Import Example – Step 6
Unwrap Symmetric Key (KEY1) Using z/TPF's Private Key and Add KEY1 to the z/TPF Keystore



© 2008 IBM Corporation

Initial z/TPF PKI Support Summary (statement of direction)

- Create and manage RSA public key pairs in a secure manner on z/TPF
  - Create, activate, deactivate, display, delete RSA public key pairs
  - Backup and restore the PKI keystore
- Use the RSA keys generated on z/TPF to create digital certificate requests as well as self-signed digital certificates
- Enable z/TPF SSL applications and middleware to use private keys generated by z/TPF
- Ability to extract a public key from the z/TPF keystore that can be distributed to remote partners such as key managers
- Ability to import a symmetric key in a secure manner using public key cryptography

© 2008 IBM Corporation

z/TPF PKI Support Phase 2 (statement of direction)

- **New operator command to display contents of a digital certificate in human-readable format**
- **Allow applications and middleware to use RSA directly:**
  - APIs to encrypt/decrypt user data using RSA public key cryptography
  - APIs to create and verify RSA digital signatures

© 2008 IBM Corporation

Questions?

- **Answers:**
  1. It depends
  2. No
  3. Yes
  4. We are exploring that
  5. Soon
  6. I'll be retired by then
  7. Go ask your mother

© 2008 IBM Corporation

Trademarks

- IBM is a trademark of International Business Machines Corporation in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.

- Notes
- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- 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.

- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.

© 2008 IBM Corporation