

# How to setup and use Keyman/VSE

Creation and upload of RSA keys and SSL certificates Creation of PFX and JKS keystores Export and import of PGP public keys

Last formatted on: Monday, July 31, 2017

Joerg Schmidbauer jschmidb@de.ibm.com

> Dept. 3252 VSE Development IBM Lab Böblingen Schönaicherstr. 220

D-71032 Böblingen Germany



## Disclaimer

This publication is intended to help VSE system programmers setting up infrastructure for their operating environment. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk. Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment. Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows XP, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

# Contents

1	Intro	oduction	5
2	Insta	alling the prerequisite programs	5
	2.1	VSE Connector Client	5
	2.2	Bouncy Castle	5
3	Initi	al Keyman/VSE setup	6
4	Som	he basic knowledge about keys and certificates	7
	4.1	Symmetric and asymmetric keys	7
	4.1.	1 RSA keys	8
	4.1.2	2 Diffie-Hellman parameters	8
	4.1.3	3 EC keys	9
	4.2	Types of SSL certificates	9
	4.3	Structure of a key ring	9
5	Rela	ationship to TCP/IP utilities	10
5	5 1	CIAI SRVR	10
	52	CIALROOT	10
	5.3	CIALCERT	
	5.4	CIALSIGV	
	5.5	CIALCREQ	
6	Ova	ruiew on key stores	12
0	Ove		12
7	Usir	ng Keyman/VSE	13
	7.1	Setting up SSL between a workstation and VSE	14
	7.2	Setting up the RSA keys for Encryption Facility for z/VSE	14
	7.3	Creating self-signed and CA-signed key rings	14
	7.4	Support for a CIALEXIT phase	14
	7.5	Activating a CIAL trace	16
8	Ove	rview of selected functions	17
	8.1	Creating keys	17
	8.2	Creating certificates	18
	8.2.	1 Using the wizard dialogs	19
	8.2.2	2 Doing it locally in Keyman/VSE	19
	8.2.3	3 Using a PRVK on VSE	19
	8.3	Using certificate mappings	20
	8.4	Displaying key material	
	8.5	Comparing keys and certificates	
	8.6	Renewing a certificate	
	ð./	Converting learnings	
	0.0	Converting Reynings	
9	Kno	wn problems	
	9.1	SSL203E RSAD failed	
10	) N	More information	29

## Changes:

November 2010 – initial version.

Sept 2011 – corrected description of CIALCREQ in section 5.5 on page 12 and drawing in section 4.3 on page 9.

Dec 2015 – added information about new features with Keyman/VSE 6.1.0: support for Diffie-Hellman parameters and Elliptic-Curve keys, new functions for comparing certificates and keys, and renewing certificates.

Aug 2016 – added info about Elliptic Curve and the Convert Keyring dialog.

July 2017 – added info on external dependency on Bouncy Castle library.

## 1 Introduction

This paper describes the setup and use of the Keyman/VSE utility. Keyman/VSE allows managing VSE specific public key infrastructure. This includes the creation of RSA key pairs and SSL certificates, which can be uploaded to VSE and stored in VSE-side key rings. In addition to that, Keyman/VSE supports workstation-based key stores and importing/exporting of PGP public keys.

Keyman/VSE is intended as a System Administrator tool. Although providing wizard dialogs to create VSE key rings, it requires some basic knowledge about RSA keys, SSL certificates, and key stores.

The usage of Keyman/VSE for specific tasks is already described at many places: VSE books, Redbooks, technical papers, presentations. Some examples are given in section "Setting up SSL between a workstation and VSE" on page 14 and section "Setting up the RSA keys for Encryption Facility for z/VSE" on page 14.

This paper tries to add some information that perhaps wasn't mentioned before.

The following software has been used in the test setup.

- z/VSE 6.1.0
- TCP/IP for z/VSE 2.1
- Java 8 from Sun/Oracle
- VSE Connector Client on the workstation side
- VSE Connector Server running on VSE
- Keyman/VSE, latest version from 2017 with external dependency on Bouncy Castle

## 2 Installing the prerequisite programs

The Keyman/VSE tool requires the installation of the VSE Connector Client and needs some Bouncy Castle jar files.

## 2.1 VSE Connector Client

You can download the most current versions of the VSE Connector Client from

http://www.ibm.com/systems/z/os/zvse/downloads/

After downloading and unpacking the zip-file, execute one of the contained install scripts:

- Setup.bat for Windows
- Setup.cmd for Windows NT
- Setup.sh for Unix / Linux

## 2.2 Bouncy Castle

Bouncy Castle is an Open Source Java class library, providing cryptography related functionality. Refer to the BC website for details (<u>https://www.bouncycastle.org/</u>).

Keyman/VSE needs two Bouncy Castle jar files that you have to download separately from <u>https://www.bouncycastle.org/latest\_releases.html</u> for legal reasons:

- bcpkix-jdk15on-152.jar
- bcprov-ext-jdk15on-152.jar

After downloading the jar files, copy them into the Keyman/VSE installation directory. When the jar files are missing, Keyman/VSE displays this message box at startup:



The names of the jar files may change with new BC releases. If the 1.5.7 versions of the jar files are no longer available, just use their successors and change your run script accordingly. The run scripts contain references to the BC jar file names:

set CLASSPATH=vkeyman.jar;bcpkix-jdk15on-157.jar;bcprov-jdk15on-157.jar; ...

## 3 Initial Keyman/VSE setup

Before you begin, read the What's new section.

🚣 K	🚣 Keyman/VSE - C:\Users\IBM_ADMIN\Documents\CONVERT1.pem *							
<u>F</u> ile	<u>Options</u>	<u>A</u> ctions	Help					
	8		⑦ General <u>h</u> elp	н	-	1		
	Label		What's new	С	Туре	VSE User	Valid	$\square$
			Select help browser	в				
			Select editor	E				
			Change properties file locatio	<b>n</b> ₽				
			🚯 About Keyman/VSE	А				
					1	VSES	SL	-

Before you can upload keys to VSE you first have to define your VSE system to Keyman/VSE.

When starting Keyman/VSE the first time, the GUI comes up with no VSE system defined. To define your first VSE system, click on the "VSE Host properties" button.

🏄 Keyman/VSE - C:	\vsecon\samples\Keyring.p	fx *			
File Options Action	ıs <u>H</u> elp				
	K 💀 😐 🛛 🕅 K 😽 G	K 💷 🗔	➡ ➡	0	
Label	VSE Host properties	Length	Туре	VSE User	Valid
				JAMPLL	

On the VSE Host – Properties dialog box press **New** to define a new VSE host. Then enter the VSE properties like IP address, VSE user and password and so on.

VSE Host - Properties		
Name	VSESSL	New
IP Address	vsessi	Add
Port	2893	Delete
VSE User	JSCH	Change
	LDAP Signon (since z/V	SE 5.1)
VSE Job Class	A	
VSE Password	*****	**
VSE Crypto Library	CRYPTO KEYR	RING
Cert. Member Name	TEST01 . PRV	K / CERT / ROOT
Cert. Mapping Member	BSSDCUID . MAP	PING
TCP/IP Library	PRD2 TCPI	PC
TCP/IP System ID	00	
		OK Close Help

Press Add to add this definition to the Keyman/VSE configuration and continue adding further VSE systems or press OK to add this definition and leave the dialog box.

Now start the VSE Connector Server on the VSE side.

You may immediately check the connection by pressing the "red light button" left to the hosts drop-down box:

🚣 Keyman/VSE - C:\vsecon\samples\Keyring.pfx *							
<u>File</u> Options	Actions <u>H</u> elp						
┢ 🖪 🖪	🖹 💀 🔳 🔺	×		⇒ ⇒	3		
Label	Certificate Item		Length	Туре	VSE User	Valid	
VSESSL is now (	connected.				VSESSL	-	

If the light turns to green, the VSE Connector Server is now connected.

## 4 Some basic knowledge about keys and certificates

This chapter provides some Keyman/VSE related overview on RSA keys and certificates. For a more detailed description on cryptography on z/VSE, refer to z/VSE TCP/IP Support, SC34-2706 and IBM Redbook Security on IBM z/VSE, chapter 4.

## 4.1 Symmetric and asymmetric keys

Keyman/VSE 6.1.0 supports the creation and management of two types of asymmetric keys:

• RSA keys

• Elliptic-Curve (EC) keys

Keyman/VSE does not support the creation of symmetric keys. Symmetric keys are created implicitly when opening an SSL/TLS connection. OpenSSL on z/VSE transparently uses crypto hardware (CCA coprocessors or CPACF) for generating random bytes to be used as symmetric session keys.

### 4.1.1 RSA keys

The term "RSA" is based on the initials of the three inventors of the RSA encryption algorithm: Ron Rivest, Adi Shamir, and Leonard Adleman. They were researchers at the M.I.T. and first described the algorithm in 1978.

RSA keys are so called "asymmetric" keys, because one key is used for encrypting data while a second key is used for decrypting. In contrast to RSA, a second type of digital key is called "symmetric" key. With symmetric keys encryption and decryption of data is performed using the same key. Examples of symmetric encryption algorithms are DES, Triple-DES, and AES.

Usually, RSA keys are not used for encrypting huge amounts of data because of their high computational effort. Instead, data is encrypted using a symmetric encryption algorithm, then the symmetric key, which just consists of a few bytes (normally 8 up to 32 bytes), is encrypted with an RSA key. The encrypted symmetric key is then put together with the encrypted data. This technique is for example used by Encryption Facility for z/VSE (refer to IBM Redbook "Security on IBM z/VSE").

When using SSL, RSA encryption is used when opening a connection. After establishing the connection, data is symmetrically encrypted.

An RSA key pair consists of two keys: the public key and the private key. The public key consists of the public exponent and the public modulus. The private key can exist in two different formats, the Modulus-Exponent (ME) format and the Chinese-Remainder-Theorem (CRT) format. In the private ME form, the private key consists of the private exponent and the public modulus, in the CRT form, it consists of five CRT parts: first prime p, second prime q, inverse first prime dp, inverse second prime dq, and CRT coefficient U and the public modulus.

The ME format is usually used when encrypting information, while the CRT format is used for decrypting.

The following list shows this structure.

Public key:

- Public exponent (e)
- Public modulus (n)

Private key:

- Private exponent
  - CRT-form: p, q, dp, dq, U
  - ME-form: private exponent (d)
- Public modulus (n)

Note that the public modulus is part of both key parts: the public key and the private key. You will see these key parts when displaying the settings of an RSA key in Keyman/VSE (see section "Displaying key material" on page 21).

For more detailed information and mathematical background refer to the "RSA" section on Wikipedia:

http://en.wikipedia.org/wiki/Rsa

### 4.1.2 Diffie-Hellman parameters

Diffie-Hellman (DH) parameters consist of two numbers p (a large prime number) and g (the generator value, which is always 2 for OpenSSL). DH parameter generation is CPU expensive, and is therefore not done during the SSL/TLS handshake process. With the help of the DH parameters, the secret symmetric session key can be calculated on both sides without being sent over the network.

### 4.1.3 EC keys

Elliptic Curve Cryptography (ECC) is an encryption technique that provides public-key encryption similar to RSA. While the security strength of RSA is based on very large prime numbers, ECC uses the mathematical theory of Elliptic Curves and achieves the same security level with much smaller keys. EC keys are based on Elliptic Curves, i.e. a specific EC key is based on a specific Elliptic Curve.

The use of EC keys together with Diffie-Hellman parameters allows the use of additional SSL/TLS cipher suites prefixed with DHE-RSA and ECDHE-RSA. Keyman/VSE 6.1.0 supports the creation and upload of EC keys to a z/VSE system. DHE-RSA and ECDHE-RSA cipher suites can only be used with OpenSSL on z/VSE.

## 4.2 Types of SSL certificates

In general there are two types of SSL certificates:

- **Root certificates** belong to a Certification Authority (CA) and may be self-signed or signed by an official CA like Thawte or Verisign. They are called *private* when they contain a private/public key pair. They are called *public*, when they only contain a public key. Private root certificates themselves can be used to sign user certificates, establishing trust.
- User certificates are signed by a CA root certificate and are a vehicle to securely transport a public key over a network.

Keyman/VSE allows creating self-signed root certificates for testing purposes in a closed environment. Because they are not signed by an official CA, no one outside your test environment would trust them. Keyman/VSE also allows creating of CA-signed root certificates.

Keyman/VSE also allows creating certificate requests that can be signed by an external CA.

## 4.3 Structure of a key ring

A complete VSE key ring consists of an RSA key pair (the PRVK member), a root certificate (the ROOT member), and a user certificate (the CERT member). Below picture shows how they are related to each other.



The above steps to create a VSE key ring are supported in Keyman/VSE through wizard dialogs; however, each step can also be performed manually. The term "subject" always means the person or site who requests an SSL certificate, while the term "issuer" means the authority which signs a certification request.

# 5 Relationship to TCP/IP utilities

Keyman/VSE uses utilities provided by TCP/IP for VSE/ESA in order to upload keys and certificates to VSE. These utilities are primarily intended to be used manually, but are used internally by Keyman/VSE.

## 5.1 CIALSRVR

The CIALSRVR utility is started by Keyman/VSE when uploading an RSA key pair. It is used to receive encrypted key material from a CIAL client program and to store the RSA key in a VSE library member with member type PRVK. Keyman/VSE hereby acts as the CIAL client, i.e. implements the CIAL client/server protocol. Another CIAL client is provided by CSI and can be downloaded from the CSI website.

CIALSRVR is started with a JCL similar to the following.

```
// JOB CIALSRVR
// OPTION SYSPARM='00'
// LIBDEF PHASE,SEARCH=(IJSYSRS.SYSLIB,PRD1.BASE)
// EXEC CIALSRVR,SIZE=CIALSRVR,PARM='CRYPTO.KEYRING.TEST01'
SETPORT 6045
/*
/&
```

When started, CIALSRVR listens on the specified port to receive the RSA key material sent from a CIAL client. After storing the key in a PRVK member, CIALSRVR ends.

## 5.2 CIALROOT

The CIALROOT utility is used to upload a CA-root certificate to VSE and catalog it as a VSE library member with member type ROOT. Keyman/VSE uses JCL similar to the following.

```
// JOB CIALROOT
// OPTION SYSPARM='00'
// LIBDEF PHASE, SEARCH=(PRD1.BASE)
// EXEC CIALROOT,SIZE=CIALROOT,PARM='CRYPTO.KEYRING.TEST01'
----BEGIN CERTIFICATE-----
MIICsDCCAhkCBB89tb0wDQYJKoZIhvcNAQEFBQAwgZ4xHjAcBgkqhkiG9w0BCQEW
D3p2c2VAZGUuaWJtLmNvbTELMAkGA1UEBhMCREUxGzAZBqNVBAqTEkJhZGVuLVd1
ZXJ0dGVtYmVyZzETMBEGA1UEBxMKQm91YmxpbmdlbjEMMAoGA1UEChMDSUJNMRQw
EqYDVQQLEwtJQk0qR2VybWFueTEZMBcGA1UEAxMQUHJpdmF0ZSBLZXkqQ2VydDAe
Fw0xMDA5MTcxMDI5MTdaFw0xNTA2MTcxMDI5MTdaMIGeMR4wHAYJKoZIhvcNAQkB
HJ9yorVUeKrvZex2KD8VnDse0/fE98Y2CE255aeVRysozO6KVpYtqksbFL11IhzG
KlKy08BTiqvh2ZJZZT0TDFFPPrM6nmKT2Y9dqz6i0kKpIkQZ8GcT+oEqiJTq29tB
3wnt+E0jVFzvnprlAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEABPNowitNtANW7rPT
DOGgyv8GovEc5+oF+hP/LmHVaWQi7fArywCeqGPl2AAxtk68bY7rprNh8IP26ezw
X4qva7DeTi6fbLIHW/x7xijmHjqiiX+o+qMcDB6jjuxqKQmMb9hSfnmj/FVBCTJf
irdlc1givE1Cu7t/rupwliqEPhs=
----END CERTIFICATE----
/*
/&
```

The certificate data is given in its Base-64 encoded portable text form. A root certificate may be self-signed or signed by an official Certificate Authority (CA) like Thawte or Verisign.

## 5.3 CIALCERT

The CIALCERT utility is used to upload a user certificate to VSE and catalog it as a VSE library member with member type CERT. Typically, a user cert contains the public key part of the related PRVK member and is signed by the related ROOT certificate.

CIALCERT is used with JCL similar to the following.

```
// JOB CIALCERT
// OPTION SYSPARM='00'
// LIBDEF PHASE, SEARCH=(PRD1.BASE)
// EXEC CIALCERT, SIZE=CIALCERT, PARM='CRYPTO.KEYRING.TEST01'
----BEGIN CERTIFICATE----
MIICpjCCAq8CBB9OyGqwDQYJKoZIhvcNAQEFBQAwqZ4xHjAcBqkqhkiG9w0BCQEW
D3p2c2VAZGUuaWJtLmNvbTELMAkGA1UEBhMCREUxGzAZBqNVBAqTEkJhZGVuLVd1
ZXJ0dGVtYmVyZzETMBEGA1UEBxMKQm91YmxpbmdlbjEMMAoGA1UEChMDSUJNMRQw
EqYDVQQLEwtJQk0qR2VybWFueTEZMBcGA1UEAxMQUHJpdmF0ZSBLZXkqQ2VydDAe
Fw0xMDA5MTcxMDQ3NTdaFw0xMTA5MTcxMDQ3NTdaMIGUMR4wHAYJKoZIhvcNAQkB
. . .
jKFxoWqF04OhJPVOb3vy0Rq2KJOoNm160byHNRq0WBbF1jcpOqOUTHppLjA81zFj
e9qAC2j+bGbK4X2/qNd89YMH4zNjQdr24QNLQ/KTA4/kH0QXMj0/E/5iqtGk188Z
MPMCAwEAATANBgkqhkiG9w0BAQUFAAOBgQBEeREKsunbLAHCo36IN8N1K2Ly7rf6
yaABtu01z77jw3K1aRlsVp9/Rynzi57jKVvh80w4E0G87IUCZKwb8RUmt119BC3i
852CEKo1X6zE0JYlnqF73oXVMuFVLJplr0XPY1uCfYdv8RYeAYyGEVOi0L7hY99W
m0JiBdh8Dwa1AA==
----END CERTIFICATE-----
/*
/&
```

### 5.4 CIALSIGV

The CIALSIGV utility is used to verify the signature of a VSE keyring. A keyring consists of three VSE library members with member types

- PRVK: contains the RSA key pair.
- CERT: contains the public key of the PRVK together with some personal information and a signature created by the ROOT certificate.
- ROOT: is either a self-signed or CA-signed root certificate.

After creating a complete VSE keyring, CIALSIGV can be used to verify its correctness. CIALSIGV is used with JCL similar to the following.

```
// JOB CIALSIGV VERIFY SIGNATURE
// OPTION SYSPARM='00' SYSID OF MAIN TCP/IP PARTITION
// LIBDEF *,SEARCH=(PRD2.TCP15G,PRD2.CONFIG,PRD1.BASE)
// EXEC CIALSIGV,SIZE=CIALSIGV,PARM='CRYPTO.KEYRING.TEST01'
/*
/&
$$ E0J
```

In Keyman/VSE, the CIALSIGV utility is used when selecting Actions - Validate VSE keyring.

## 5.5 CIALCREQ

The CIALCREQ utility is used to create a certificate request from a given PRVK member. Basically this means to take the public key of the member PRVK and add some user provided personal information. Then, a hash value is calculated from public key and personal information, and finally signed with the private key. The certificate request (personal information, public key, hash algorithm, and signature) is then sent to a Certification Authority (CA), which constructs a x.509 certificate from the given information. The obtained x.509 certificate is then cataloged as a CERT member on VSE. Refer to RFC 2314, which describes the syntax for certification requests in detail. Also refer to section "Using a PRVK member on z/VSE" on page 19 for how to create a certificate request from a given RSA key pair.

Restriction: CIALCREQ cannot create a certificate request from a 4096-bit key.

CIALCREQ is used with JCL similar to the following.

```
// JOB CIALCREQ CREATE CERT REQUEST
// OPTION SYSPARM='00' SYSID OF MAIN TCP/IP PARTITION
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASE)
// EXEC CIALCREQ,SIZE=CIALCREQ,PARM='CRYPTO.KEYRING.TEST01'
Common-name: www.ssl4vse.com
Organization Unit: Development
Organization: Connectivity Systems
Locality: Columbus
State: Ohio
Country: US
/*
/&
```

In Keyman/VSE, the CIALCREQ utility is used when creating a certificate request from an RSA key that is already stored on VSE in a PRVK member (pop-up menu of a root certificate – Create VSE cert via CIALCREQ).

## 6 Overview on key stores

There are many different types of key stores used on workstation platforms. However, three formats represent some common standard and are supported by Keyman/VSE.

- 1. **PFX**: The PFX (Personal Information Exchange) format was initially defined by RSA Security and conforms to the PKCS#12 standard. PFX files can contain multiple keys and certificates and are themselves password-protected. PFX files are supported by Web Browsers like Microsoft Internet Explorer and Mozilla Firefox. Sometimes the file extension p12 is also used for the PFX format.
- 2. **JKS**: The JKS (Java Key Store) format is provided by Sun Microsystems and is the standard key store format for Java applications. Also JKS files are protected by a password. JKS files usually cannot be handled by Web Browsers. Part of each Java installation is the keytool.exe, which can also be used for maintaining JKS key stores.
- 3. **PEM:** The Privacy-enhanced mail (PEM) format is used by OpenSSL. PEM files can contain an RSA key pair, an SSL certificate or both. It can but must not be password protected. Whereas other keystore formats are just binary, the PEM file content is base-64 encoded

All three formats can be used for storing SSL certificates on a workstation when setting up an SSL connection to VSE. On the VSE side, the related keys and certificates are stored in a VSE key ring consisting of VSE library members.

You can specify the type of key store when saving certificate items in Keyman/VSE.

Save local keyring file			X		
Look <u>i</u> n: samples		▼ 6 2 2 8	0-1-		
	ring.JKS NUAL.JKS TEST.JKS iest2.JKS DDLE.JKS RIPT01.JKS IVER01.JKS	<ul> <li>\$sl03.JKS</li> <li>test.JKS</li> <li>VSEC08_TEST01.JKS</li> <li>VSESSL_CTEST.JKS</li> <li>vsessl_VSESSLJKS</li> <li>VSESSLDH.JKS</li> <li>WIZ1234.JKS</li> <li>WIZ1234.JKS</li> </ul>	0		
Imploiting     Imploiting       File Name:     sample.jks       Files of Type:     JKS (Java Key Store)	e)				
PFX Options       JKS Options       PEM Options         JKS file password       *******					
		Save Cancel	lelp		

PGP (Pretty Good Privacy) uses another key store format (KDB: Key Data Base) that is not supported by Keyman/VSE, however, section "Setting up the RSA keys for Encryption Facility for z/VSE" on page 14 shows that Keyman/VSE can make PGP public keys usable for VSE.

## 7 Using Keyman/VSE

In general you will use Keyman/VSE for two main tasks: setting up SSL between a workstation and VSE, and setting up the RSA keys for Encryption Facility for z/VSE.

### 7.1 Setting up SSL between a workstation and VSE

Setting up SSL in various VSE environments is described in detail in IBM Redbook "Security on IBM z/VSE", available online at

http://www.redbooks.ibm.com/abstracts/sg247691.html?Open

This includes setting up SSL for

- CICS Web Support (CWS)
- Secure Telnet
- Secure FTP
- WebSphere MQ for z/VSE
- SSL with z/VSE e-business connectors

Also, refer to other technical papers on the VSE homepage:

http://www.ibm.com/systems/z/os/zvse/documentation/security.html#howto

## 7.2 Setting up the RSA keys for Encryption Facility for z/VSE

Encryption Facility for z/VSE is an optional priced feature available for z/VSE 4.1 and later. It provides encryption of tapes (real tapes and virtual tapes) and VSE datasets on disk (SAM files, VSAM files, VSE Library members). Encryption Facility requires the CPACF (CPU Assist for Cryptographic Function, feature code #3863), i.e. runs on a z890 or higher.

Encryption can be done either by specifying a password or by using an RSA key to protect the encryption key used for encrypting the data. When RSA public-key encryption is used, Keyman/VSE allows creating and uploading the necessary RSA keys to VSE.

Encryption Facility for z/VSE V1.2 provides support for the OpenPGP message format (refer to RFC 4880). Therefore, support for importing PGP public keys and converting them into the x.509 certificate format has been added to Keyman/VSE. This is necessary for exchanging PGP public keys with other platforms, such as workstations and z/OS.

This is described in detail in IBM Redbook "Security on IBM z/VSE", available online at

http://www.redbooks.ibm.com/abstracts/sg247691.html?Open

Refer to Chapter 4.6 "Software-based encryption with Encryption Facility for z/VSE V1R2".

## 7.3 Creating self-signed and CA-signed key rings

The Keyman/VSE help section contains detailed descriptions of how to create VSE-side key rings. Open the Keyman/VSE "General Help" and click on the "How to..." link.

## 7.4 Support for a CIALEXIT phase

With the currently latest version of Keyman/VSE from Oct. 2010, support for a CIALEXIT phase is provided. A CIALEXIT phase allows defining a custom passphrase that must be entered whenever uploading a private key to VSE via the CIALSRVR utility.

When not using a CIALEXIT, a CIAL client encrypts the created binary key material with a hard-coded Triple-DES or AES key. A SHA-1 hash of a hard-coded passphrase is appended to the key material and sent to CIALSRVR, which verifies the passphrase hash before writing the key into a PRVK member.

With a CIALEXIT, this process gets much more secure, because the passphrase and involved encryption keys are under full control of the customer. A sample CIALEXIT program is provided with the Keyman/VSE installation package.

If a CIALEXIT phase is cataloged on VSE, CIALSRVR enforces Keyman/VSE to display a password prompt. The entered passphrase is then SHA-1 hashed and sent to CIALSRVR, which verifies the password hash with the password in CIALEXIT.

The following dialog sequence results.

🛓 Keyman/VS	🚣 Keyman/VSE - C:\vsecon\samples\Keyring.pfx *							
File Options	<u>A</u> ctions <u>H</u> elp							
🧀 🖪 📭	🖹 📑 🛃	×¢	( 🔜 🔜	⇒ ⇒	8			
Label	Certificate Item		Length	Туре	VSE	User	Valid	
Ҟ vseKey	2048-bit RSA Key Pair	Vie	w (Text)	DA1D * / 1/	_	-		
		Sof	tinge					
			unys					
		A Del	ete		Delete			
		🔓 Upl	oad to VSE					
		🖄 C <u>r</u> e	ate VSE serv	er cert request				
		Exp	oort text form	)		1		
		Exr	oort binary fo	rm				
			ov to clinhoar	'n				
			ort on DCD n	ublic kov				
		EX	2011 as PGP p	ирис кеу		ESSL	-	

First, you create an RSA key and select "Upload to VSE".

Send certificate iten	n to VSESSL		X		
Crypto Library Member Name	CRYPTO TEST01	. KEYRING	<b>•</b>		
Make sure that the V	SE Connector Ser	ver is started in no	on-SSL mode. 🔺		
			-		
		Upload Cl	ose Help		

#### Press Upload.

Private key passphrase needed					
CSpecify Passphrase for CIALSRVR					
•••••					
A CIALEXIT phase is cataloged on the VSE side specifying the private key password. The same password must be specified					
here and is used by CIALSRVR when cataloging the private key.					
OK Cancel					

The "Private key passphrase needed" dialog box is displayed. Here enter the secret passphrase specified in CIALEXIT.

Send Certificate Item To VSE					
VSE Host	H05:9.152.109.41				
Crypto Library	CRYPTO . KEYRING				
Member Name	TEST01 . PRVK 💉				
Prompting for secret passphrase for CIALEXIT phase Private key passphrase specified. Sending private key					
Finished.					
	Upload Close Help				

Press **Close** to end the procedure.

When using a CIALEXIT phase you have to consider the following.

- CIALEXIT not only requests a secret passphrase from the user to be allowed for storing the key data, but also uses the hardcoded TDES or AES key in CIALEXIT to encrypt the RSA key data in the PRVK member. This causes the key and CIALEXIT phase tied together and makes it impossible to simply punch or restore a key at another site.
- As a consequence, if the CIALEXIT phase ever gets lost or is changed accidentally, the related RSA key is no more usable.
- Regardless whether using a CIALEXIT phase or not, Keyman/VSE does encrypt the RSA key with a hard-coded TDES or AES key before sending the key blob to CIALSRVR. CIALSRVR then re-encodes the RSA key with the TDES/AES key in CIALEXIT before writing it to the PRVK member.

## 7.5 Activating a CIAL trace

If you have any problems cataloging keys or certificates, you can activate a CIAL trace. On the Keyman/VSE main window select Options – Trace settings.



On the Trace Settings box just mark checkbox CIAL Trace.

Keyman/VSE - Trace Settings	
CIAL Trace CIAL Trace Adds additional JCL lines to CIAL jo a \$SOCKDBG phase for full output.	obs. You have to catalog Press help for details.
	OK Cancel Help

#### Press OK.

The CIAL trace generates SDUMPS in the output of all CIAL utilities. Additional JCL statements are added to the JCL created by Keyman/VSE:

// OPTION NOSYSDMP // UPSI 1

In addition to activating the trace in Keyman/VSE, you have to catalog a \$SOCKDBG phase for full list output.

```
// JOB $SOCKDBG
// OPTION CATAL
// LIBDEF *, CATALOG=PRD1.BASE
// EXEC ASMA90, SIZE=ASMA90
                  ' PHASE $SOCKDBG, * '
        PUNCH
$SOCKDBG CSECT
                                GENERATE A PHASE
        SOCKDBG CSECT,
                                                                        Х
               FL01=$DBGWLST, +DBGWLOG, MESSAGES TO SYSLST AND SYSLOG X
               FL02=$DBGISON,
                              DEBUG IS ON
                                                                        Х
               FL03=$DBGNONE, NONE
                                                                        Х
               MSGT=$DBGALL,
                                ISSUE ALL DIAGNOSTIC MESSAGES
                                                                        Х
                                NO DIAGNOSTIC SDUMPS FOR IPNRBSDC
                                                                        Х
               DUMP=$DBGNONE,
               SSLD=$DBGSDMP,
                                YES DIAGNOSTIC SDUMPS FOR IPCRYPTO
                                                                        Х
               CIAL=$DBGSDMP,
                                YES DIAGNOSTIC SDUMPS FOR IPDSCIAL
                                                                        Х
                                NO DIAGNOSTIC SDUMPS FOR CIALCECZ
               CECZ=$DBGNONE
               $SOCKDBG
        END
/*
  EXEC LNKEDT, SIZE=512K
//
/*
/&
```

## 8 Overview of selected functions

This chapter adds some information about Keyman/VSE that might not be explicitly mentioned in other books.

## 8.1 Creating keys

Keyman/VSE provides two functions for generating RSA keys.

1. A key can be created locally in Keyman/VSE using Java functionality. The key must then be uploaded to VSE in a separate step. Although the key material gets encrypted before sending it over the network, this is clearly a security exposure. You may consider using a CIALEXIT phase in this case (refer to section "Support for a CIALEXIT phase" on page 14).

Generate new RSA key	
Local Mainframe	
Key length	2048 💌
Alias	vseKey
Notes: This function creates a new loo key must then be uploaded to V member. Key lengths greater t hardware on the mainframe fo	cal RSA key in Keyman/VSE. This /SE and cataloged in a PRVK han 1024 require cryptographic r using such keys later.
	Generate key Close Help

- 2. A key can be created directly on the mainframe via the CIALSRVR utility. This function requires cryptographic hardware on the host side:
  - Creating keys up to 2048-bit key length requires a Crypto Express2 card
  - Creating keys with 4096-bit key length requires a Crypto Express3 card

Generate new RSA key	
Local Mainframe	
Keyring library	CRYPTO . KEYRING
VSE Library Member	TEST02 . PRVK
Key length	2048 💌
Notes:	
This function generates a ne	w RSA key directly on the mainframe
using a Crypto Express CCA	coprocessor card. Crypto Express2 cards
can create keys up to 2048 b	its, Crypto Express3 and later can create
keys up to 4096 bits. This fun	action requires ZP15F411 up to ZP15F419.
View job output	
	Generate key Close Help

This function uses the new command GENRSAPK provided by the CIALSRVR utility.

For more information about System z cryptographic hardware refer to

http://www.ibm.com/systems/z/security/cryptography.html

Note that the usage of keys greater than 1024 bits always requires a crypto card on the host side. TCP/IP for VSE/ESA only provides a software implementation of the RSA algorithm for 512 and 1024 bit keys.

## 8.2 Creating certificates

Actually there are three ways of creating a certificate on the basis of a given RSA key, some personal information to be specified by the user, and a root certificate used to create the signature.

### 8.2.1 Using the wizard dialogs

The easiest way of creating a complete VSE key ring is using the wizard dialogs. Pressing the green arrow button creates a self-signed key ring, i.e. uses a self-signed root certificate. Pressing the yellow arrow button allows interacting with an external Certificate Authority (CA), like Thawte or Verisign.

## 8.2.2 Doing it locally in Keyman/VSE

This option requires some knowledge about the relationship between RSA key, certificate request, and root certificate, but allows manipulating all items manually. All actions are performed locally in Keyman/VSE. There are no CIAL utilities used.

The steps are:

- 1. Create a new RSA key
- 2. Create a self-signed root certificate
- 3. Create a certificate request with the public key from the RSA key pair.
- 4. Sign the certificate request with the root certificate.
- 5. Delete the certificate request.

The steps are described in detail in the Keyman/VSE "How to" help section.

### 8.2.3 Using a PRVK on VSE

While the two above alternatives assume the RSA key being present in Keyman/VSE, this option allows using a remote PRVK for creating the certificate request.



Select option "Create VSE cert via CIALCREQ".

On the next dialog box, the PRVK to be used can be specified.

**Restriction**: this function does not work with a 4096-bit key, because the CIALCREQ utility cannot handle this key length. Use the native Keyman/VSE function instead (pop-up menu of an RSA key – Create VSE server cert request).

Personal information	n for X509 user certificate
Common name	Test Server Certificate
Organizational Unit	Development
Organization	IBM
City/Location	Boeblingen
State/Province	Baden-Wuerttemberg
Country	DE Germany (DE)
e-mail	zvse@de.ibm.com
Expires	2017-12-18 2 years 💌
Alias	userCert
Key length	Taken from PRVK on host
Public key	CRYPTO . KEYRING . TEST01 . PRVK
Notes: Make sure that the CIALCREQ does no	• VSE Connector Server is started on VSESSL ot work with a 4096-bit key!
Generate ce	ertificate Close View output Help

Press Generate certificate.

## 8.3 Using certificate mappings

Keyman/VSE supports mapping of SSL client certificates to VSE user IDs. This is mainly used by CICS Web Support (CWS) together with SSL client authentication. When a CWS SSL client tries to invoke a CICS transaction, the mapping between SSL client certificate and VSE user ID allows deciding whether to grant or deny access.

The VSE Java-based connector also supports certificate mapping. The VSE Navigator application is an example for using certificate mapping in order to logon to VSE without being prompted for a password.

Mapped SSL client certificates are not uploaded to VSE via CIAL utilities. Instead, the BSSDCERT utility provided by the Basic Security Manager (BSM) is used. How to use BSSDCERT manually is described in the z/VSE Administration Guide. However, Keyman/VSE can create and submit all jobs automatically. From the GUI perspective there is no difference between uploading mapped and unmapped certificates. Just select "Upload to VSE" from the pop-up menu of the certificate.

Keyman/VSE - C:\vsecon\sam	ples\Keyring.pfx *			
File Options Actions Help				
🗀 🖺   📑 🏭 🕵 😐	Ҟ 🖬 🖬   📫	•		
Alias Certificate Item	Length	Туре	VSE User	Valid
🎋 vseKey 1024-bit RSA Key Pai	r 1024	Key Pair	-	-
🗾 rootCert 🛛 Private Key Cert	1024	ROOT	-	Yes
userCert JSCH	Settings Delete	User	-	Yes
	Export text form Export binary form Copy to dipboard	_		
	Upload to VSE Map to VSE User	-		
	Export PGP public key	l I I	H05:9.152	.109.41

On the next dialog box enter the VSE user ID to which this client certificate shall be mapped.



After uploading a mapped certificate to VSE, the mapping can be displayed via the "Client certificate mapping" function.

🕌 Keyman/V	/SE - C:\vsecon\samples\Keyring.p	fx *			
File Options	Actions Help				
┢ 🖺	📃 🐘 🛐 😐 🛛 🛝 📼 🗉	I   📫	-		
Alias	Certificate Client certificate mapping	Length	Туре	VSE User	Valid
Ҟ vseKey	1024-bit RSA Key Pair	1024	Key Pair	-	-
🔜 rootCert	Private Key Cert	1024	ROOT	-	Yes
🗔 userCert	JSCH	1024	Client	JSCH	Yes
				H05:9.152	109.41

The following dialog box basically shows the same information as the Interactive Interface dialog "MAINTAIN CERTIFICATE - USERID LIST" (fast path 2.8.4).

🛓 VSE Client Certif	icate Mapping		X
Member Name JSCH	Mapped User JSCH	Trusted YES	Subject's Common Name And Organization TEST01 SERVER CERTIFICATE, IBM
Right-click an entry i	for options.		
			Close View output Help

The next section describes some advanced functions that are normally not needed when just maintaining keys and certificates.

## 8.4 Displaying key material

The key parts of an RSA key or SSL certificate can be displayed and exported via the pop-up menu "Settings" function. This can help comparing the key parts of different keys or certificates for debug purposes. Just double-click an item in the Keyman/VSE main window to open the Settings dialog box.

For certificates, the public and private key parts can be displayed separately.

X.509 Root Certificate Setti	ngs
Issuer name :	CN=Test SSL CA Certificate, OU=IBM, O=I
Subject name :	CN=Test SSL CA Certificate, OU=IBM, O=II
Type :	Private
Serial no. :	1450447160021 Hex: 0151B56342D5
Key algorithm :	RSA 2048-bit
e-mail :	zvse@de.ibm.com
Has public key :	Yes
Has private key :	Yes
Valid :	Yes
Valid from :	Fri Dec 18 14:59:19 CET 2015
Valid to :	Thu Sep 05 14:59:19 CEST 2019
Signature algo :	SHA256WithRSAEncryption
Signature :	
AE B8 F6 97 DE 63 A7	0B 65 A5 8E 1D 5A F3 39 49
Certificate Alias	Advanced Show public key Show private key
	Change Cancel Help

For example, press the **Show private key** button.

💁 Private key material	
BEGIN RSA PRIVATE KEY	
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCzDcxlUnBs	
GX0vwMJP9HC31wscYg1siWuH5369on6nnDxXMxXVIC57GLe128dsDi70YFdj	
8X8u6+8f03znp4n3A7VE1Bthm4JJHG6x6T75JpF6Lqjuk07HiZk0B6C8631j	_
e8UHxrxr1ki19z06IOjVG6FR2WZKguVYpdcXibSomJ19QbSp2Cw0eW2aMBw1	-
AEi04LGJ68fSyCb0GnUMaG3ibcy5Z2faaP3wNs5WC+n2xryYmuYYrXLAG8Kr	
d9VJe12MBMSCOcDYINMod+jegaDXSuyWUZWUBYemjVszR5ank+fHxbnO44DN	
014T5CuFC1QJ1Aejvfz7x9DsnnpSRrJNAgMBAAECggEAKlbj8z2YOeeZE5ao	
HBH4KXCfPvnspLKmC7/w11S6UFYAaro4umnes9r3ai/bKzD7Xm+nmol7gpPl	
uofe7ERQumlVDs59HwqPt0wS3N0DuBSnp9Wquqbo36Qo3gltt96SmLSmtZ0H	
5xQHo4ewIbOwetwKvHkE2C+1nc2xW+6DGq59kMY/J851I0+fiZJx6501S7Ut	
seXmHfd5TGFevz60GeAscySi0BJHIkUz3dKSjwyK/Kx24gx7U5iMGvIiaR5P	
QvlXx1FhE/Y98zRKh+b/bltNH9MtnG7KzMYZs94GyA/10qYZLqDEMedoS8di	
t9DsNDhj5jiRtG7KFlxc1SxSQQKBgQDhk8y84kyEdgjVvqXEZyYI7vU3+bAR	
VIE4aSauz+AaM/hvovrAHCiARNTrwSiTDVXH+AMLaCOoealFDaSazDEBYpkL	-
Copy to clipboard   Show base64 text form  Show bytes  Show number	s
Close	

The displayed information can be copied to the clipboard.

## 8.5 Comparing keys and certificates

Sometimes it is useful to compare two keys or certificates in order to verify the correctness of your setup. Keyman/VSE 6.1.0 provides a dialog to compare two keys or certificates.

Select the two items in the Keyman/VSE main window and select Compare.

4	Keyman/VS	E - C:\vsecon\samples\Keyring.pf	x *			
Eil	e <u>O</u> ptions	Actions Help				
	) 🖺 🖪	🛛 🖹 🖳 🔳 🔺 👫 😫	( 🔜 🗔		0	
	Label	Certificate Item	Length	Туре	VSE User Valid	
	vseKey	1024-bit RSA Key Pair	1024	RSAPrivateKey		
	certreq	1024-bit Certificate Request	1024	CertRequest	-	
=,	rootCert	2048-bit ROOT Certificate	2048	Compara	Yes	
Ξ.	vseCert	1024-bit User Certificate	1024	<u>Compare</u>	Yes	
				💢 <u>D</u> elete Delete	e	
					S35LP//	

In this example, the certificate request was created from the RSA key, so both items have the same public key.

Compare certificate i	tems	X
RSA Private Key	X Not available on right side	
RSA Public Key	X Different	
Issuer	V Match	•
Subject	X Different	
Validity period	X Different	
Serial number	X Different	•
Signature algorithm	V Match	
Signature	X Different	•
	Close	

Click on one of the push buttons on the right to see comparison details:

🖆 Comparison details	×
Upper item	
SHA256WithRSAEncryption	
Loweritem	
SHA256WithRSAEncryption	
Close	

## 8.6 Renewing a certificate

Sometimes it even might be helpful to extend the validity period of a certificate or recreate it with a different signature algorithm. This is now supported by dialog Renew certificate.

🛓 Keyman/VS	E - C:\vsecon\samples\Keyrii	ng.pfx *			. 🗆 🗙
File Options	<u>A</u> ctions <u>H</u> elp				
	🖹 🗟 🛛 🖹	€ 🔜 🗔	▶ ▶	0	
Label	Certificate Item	Length	Туре	VSE Us	er Valid
Not the section of th	1024-bit RSA Key Pair 1024-bit Certificate Request	1024 1024	RSAPrivateKey CertRequest	-	-
5 rootCert	2048-bit ROOT Certificate	View (Text)	D (0 (		Yes
🖬 vseCert	1024-bit User Certificate	vie <u>w</u> (Text)			Yes
		<u>S</u> ettings			
		💢 <u>D</u> elete		Delete	
		Export text f	orm		
		Export binary	y form		
		L Copy to clipb	oard		
		🚯 Upload to VS	E		
		Create VSE of	certificate		
		Create VSE o	cert via CIALCREQ.		
		Create user	certificate		
		Sign certifica	ate request		
		Export as PG	P public key		
		<u>Renew</u>			
			45	<b>a</b> 300	

Right-click a certificate and select **Renew**.

Specify new expiration date Current expiration date (YYYY-MM-DD)	2019-09-05
New expiration date	2020-9-5 <b>1 year</b>
Specify new signature algorithm	
Current signature algorithm	SHA256WithRSAEncryption
New signature algorithm	SHA-256 SHA-1

Renewing a certificate makes a copy of the certificate with all properties like private/public keys, issuer and subject information, serial number and so on. However, when changing the signature algorithm, a new signature is created

The following prerequisites must be fulfilled when renewing a certificate:

- 1. A self-signed private certificate can be renewed without additional information, because the new signature can be created using its private key.
- 2. Renewing a user certificate *with an own private key* requires a new signature from the original CA certificate that was used to sign the certificate before. Therefore, Keyman/VSE just creates a new Certification Request that can then be signed by a CA in a separate step.
- 3. A user certificate *without an own private key* cannot be renewed unless the original private/public key pair is available in Keyman/VSE. If this private/public key pair is available, a new Certification Request is created with a signature from the private key.

## 8.7 Renewing a certificate using a host-side PRVK

As said in the previous section, a user certificate *without an own private key* cannot be renewed unless the original private/public key pair is available in Keyman/VSE. However, if the original RSA key is available on a VSE system as a PRVK member, you can either download this key into the Keyman/VSE tool, or let Keyman/VSE search the host-side keyring library for a matching RSA key.

Let's assume you have a user certificate and the related CA root certificate in Keyman/VSE, but the corresponding RSA key that was used to create the user certificate's request, is no longer available locally.

Keyman/VSE - C:\Users\IBM_ADMIN\Documents\CONVERT1.pem *							
File Options	<u>File Options Actions Help</u>						
📄 🖪 🖪	► □ ► ▼ ■ ► ★ ★ ■ ■ ★ ★						
Label	Certificate Item	Length	Туре	VSE User Va	lid		
E rootCert	1024-bit ROOT Certificate	1024	SelfSigned	- Yes			
vseCert	View (Text)	1024	UserCert	- Yes			
	Pload to VSE						
	Export text form						
	Export binary form						
	Add private key						
	Change <u>s</u> erial						
	Copy to clipboard						
Map to VSE User							
	<u>√ R</u> enew ∖						
	X Delete Delete						
	Properties			VSESSL	-		

Right-click the user certificate and select Renew...

Renew X.509 certificate	
1. Specify new expiration date	
Current expiration date (YYYY-MM-DD)	2034-04-22
New expiration date	2035-04-22 Unchanged 💌
Root certificate for signing	rootCert
Original private key	▼
Search keyring lib (requires ZP15F27	5)
2. Specify new signature algorithm	
Current signature algorithm	SHA-256
New signature algorithm	● SHA-256 ○ SHA-1
3. Specify new basic constraints	
Certificate shall be a CA cert	
Number of intermediate CAs -1	
	Copy cert with new settings Close

The **Renew X.509 certificate** box already shows your local CA root certificate to be used for applying the signature to the renewed certificate. Now press the **Search keyring lib** button to scan the host-side keyring library for matching PRVKs.

Renew X.509 certificate	
1. Specify new expiration date	
Current expiration date (YYYY-MM-DD)	2034-04-22
New expiration date	2035-04-22 Unchanged 💌
Root certificate for signing	rootCert
Original private key	TESTKEY.PRVK
Search keyring lib (requires ZP15F275)	100
2. Specify new signature algorithm	
Current signature algorithm	SHA-256
New signature algorithm	● SHA-256 ○ SHA-1
3. Specify new basic constraints	
Certificate shall be a CA cert	
Number of intermediate CAs -1	•
1 key(s) inserted into dropdown box.	
C	opy cert with new settings Close

Matching keys are now displayed in the **Original private key** drop-down list box. You may now change the expiration date, and/or signature algorithm, and/or basic constraints. Finally press the **Copy cert with new settings** button.

Keyman/VSE - C:\Users\IBM_ADMIN\Documents\CONVERT1.pem *						
<u>File Options Actions Help</u>						
		( 💷 💷 🕯				
Label Ce	ertificate Item	Length	Туре	VSE U	ser Valid	
rootCert 1024-bit ROC	OT Certificate	1024	SelfSigned	-	Yes	
vseCert Test Server	Certificate	1024	UserCert		Yes	
-						
F	Renew X.509 certificat	te				X
	1. Specify new expira	ation date —				
	Current expiration da	ate (YYYY-MM	-DD)	2034-04-22		
	New expiration date			2035-04-22	Unchange	d 🖵
	Root certificate for s	ianina				
		ginig		rootCert		
	Original private key			TESTKEY.PRV	/К	•
	Search keyring lib	(requires ZF	P15F275)			
	2. Specify new signa	ture algorith	m			
	Current signature alg	jorithm		SHA-256		
	New signature algori	thm		● SHA-256 🤇	SHA-1	
	3. Specify new basic	constraints				
	Certificate shal	l be a CA cer	t			
	Number of interme	diate CAs -1			-	
	Cortificato rocomo	4				
		J.	Cor	ov cert with new	w settings	Close
L						

The user certificate is now renewed with changed expiration period and/or signature algorithm.

## 8.8 Converting keyrings

VSE keyrings, consisting of a set of three VSE library members with member types PRVK, CERT, and ROOT, can be converted to a keyring usable by OpenSSL. The contents of the three members is extracted and saved in a PEM file.

**Note:** This function requires either

- TCP/IP for z/VSE V2.1 or
- TCP/IP for VSE V1.5F with APAR PI59039, PTFs UI37730 (5.1), or UI37731 (5.2).

On the Keyman/VSE main window, click on **Convert keyring to PEM** toolbar button.

🚣 Keyman/VSE - C:\Users\IBM_ADMIN\Documents\CONVERT1.pem *							
File Options	<u>Actions</u> <u>H</u> elp						
	🖹 🛃 🖻	🔨 💦 🨫 🤤	: 🗖 🖬	R_ 🕨		,	
Label	Certificat	e Item	Length	Convert keyri	ing to PEM EUser	Valid	
					-		
					VSES	SSL	

On the **Convert VSE keyring to PEM** dialog box, click on button **<< Fill List.** 

Convert VSE Keyring to PEM					
Keyring library Keyring to convert	CRYPTO.KEYRING	< Fill List			
1 keyring(s) added to	list.	s PEM file Close			

Complete VSE keyrings, consisting of the three members .prvk, .cert, and .root, are added to the list. You can then save each keyring as a separate PEM file by pressing the **Save as PEM file** button.

👍 Keyman/VS	SE - C:\Users\IBM_ADMIN\Documents\CONVERT1.pem *
<u>File</u> Options	Convert VSE Kevring to PEM
	Valid
Labor	Keyring library CRYPTO.KEYRING
	Keyring to convert SAMPLE   Keyring to convert
	PRVK, CERT, and ROOT retrieved, keyring validated.           Save as PEM file         Close
	VSESSL 🗸

🛃 Save PEM File					
Save in:	🗼 samples	🗸 🥝 🤌 📂 🛄 🕇			
æ	Name	Date modified	Type 🔺		
	CONVERT.PEM	22.04.2016 20:43	PEM File		
Recent Places	CONVERT.PFX	22.04.2016 20:43	Personal		
	CONVERT2.PEM	21.04.2016 19:43	PEM File		
·	CONVERT1.PEM	21.04.2016 19:42	PEM File		
Desktop	KMNEW1.PEM	21.04.2016 19:27	PEM File		
	PCOM6015-RETRIEVED.PEM	21.04.2016 19:25	PEM File		
	FINAL123.PEM	19.04.2016 21:08	PEM File		
Libraries	KMNEW256.PEM	14.04.2016 13:47	PEM File		
	ccert.cer	13.04.2016 11:02	CER File		
	KMOLD.PEM	12.04.2016 15:23	PEM File		
Computer		21.02.2010.10.54	PEA EL		
	File name: SAMPLE.PEM	•	Save		
Network	Save as type: All Files (*.*)	Cancel			

When the keyring is validated, i.e. checked for consistency, a platform dependent **Save dialog** opens.

Specify the name and location of the PEM file and press Save. You may repeat these steps for further keyrings.

## 9 Known problems

This section describes some known problems.

## 9.1 SSL203E RSAD failed

#### Symptom:

Message SSL203E RSAD failed RC=0000002E(RSADNOHC) reason=000004B2 is issued when trying to open an SSL connection.

#### Reason:

Return code RSADNOHC indicates "no hardware crypto" when performing an RSA decryption. Most likely you are using an RSA key with a key length greater than 1024 bits. TCP/IP for VSE/ESA has a software implementation of the RSA algorithm up to 1024 bits. Greater keys require a crypto card installed in your mainframe. Supported cards are currently: PCICA, PCIXCC, Crypto Express2 (CEX2C and CEX2A), and Crypto Express3 (CEX3C and CEX3A). For more information on cryptographic hardware for System z refer to

http://www.ibm.com/systems/z/security/cryptography.html

# **10 More information**

You can find more information in these books and web sites:

VSE Homepage http://www.ibm.com/servers/eserver/zseries/zvse/

Keyman/VSE tool and VSE Connector Client http://www.ibm.com/servers/eserver/zseries/zvse/downloads/

z/VSE Administration http://www.ibm.com/systems/z/os/zvse/documentation/#vse

z/VSE e-business Connectors User's Guide http://www.ibm.com/systems/z/os/zvse/documentation/#conn

Redbook: Security on IBM z/VSE, SG24-7691 http://www.redbooks.ibm.com/abstracts/sg247691.html?Open

Technical papers on VSE homepage <a href="http://www.ibm.com/systems/z/os/zvse/documentation/security.html#howto">http://www.ibm.com/systems/z/os/zvse/documentation/security.html#howto</a>

RFC 4880, OpenPGP message format <u>http://tools.ietf.org/html/rfc4880</u>

CSI International Homepage <u>http://www.csi-international.com/</u>

CSI provided CIALClient http://www.csi-international.com/download.htm Here click on link "Download SSL components"

Wikipedia article on RSA http://en.wikipedia.org/wiki/Rsa

Overview on System z cryptographic hardware http://www.ibm.com/systems/z/security/cryptography.html