

# Security and Linux on IBM Z

Lydia Parziale

Klaus Egeler

Manoj S. Pattabhiraman



**Security**

**Linux**





International Technical Support Organization

**Security and Linux on IBM Z**

December 2017

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.

**First Edition (December 2017)**

This edition applies to the IBM Z platform, Red Hat Enterprise Linux Servers v7.4, and z/VM v6.4.

This document was created or updated on January 8, 2018.

**© Copyright International Business Machines Corporation 2017. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	v
Trademarks .....	vi
<b>Preface</b> .....	vii
Authors .....	vii
Now you can become a published author, too! .....	vii
Comments welcome .....	viii
Stay connected to IBM Redbooks .....	viii
<b>Chapter 1. Why security and encryption</b> .....	1
1.1 Why security matters .....	2
1.2 Why use encryption .....	2
1.3 Pervasive encryption with the IBM z14 .....	3
1.3.1 CP Assist for Cryptographic Function .....	4
1.3.2 Crypto Express6S .....	5
1.4 Benefits of Hardware Crypto .....	6
1.5 Verification of installed LIC 3863 using the SE .....	7
<b>Chapter 2. Data security and Linux on IBM Z</b> .....	9
2.1 Data security .....	10
2.1.1 Encryption .....	10
2.2 Pervasive encryption .....	11
2.3 LinuxONE and IBM Z Cryptographic Hardware features .....	12
2.3.1 CP Assist for Cryptographic Function .....	13
2.3.2 Crypto Express6S .....	13
2.4 Overview of enabling cryptographic adapters .....	14
2.4.1 Hardware configuration: Setting up an LPAR to use Crypto .....	14
2.4.2 Configuring Cryptographic adapters in z/VM .....	15
2.4.3 Setting up Cryptographic adapter for use from Linux on IBM Z .....	15
2.5 Verification of installed LIC 3863 using the SE .....	15
2.6 The hardware and software test environment .....	16
2.7 Cryptographic software support: z/VM .....	17
2.7.1 Defining Cryptographic feature in z/VM .....	17
2.7.2 Configuring Cryptographic feature in z/VM .....	18
2.7.3 Hardware cryptography exploitation in Linux on IBM Z .....	20
<b>Chapter 3. Pervasive encryption: Data-at- rest encryption</b> .....	29
3.1 Introduction .....	30
3.2 LinuxONE Hardware-accelerated in-kernel cryptography .....	30
3.2.1 Verification of support for Hardware Cryptographic operation .....	30
3.2.2 The dm-crypt module .....	31
3.3 Data-at-Rest using dm-crypt LUKS encryption .....	33
3.3.1 Setting up cryptsetup .....	34
3.3.2 Creating a LUKS partition .....	35
3.3.3 Test for demonstrating encryption performance with hardware optimization .....	38
<b>Chapter 4. Pervasive encryption: Data in flight encryption</b> .....	41
4.1 Preparing to use OpenSSL .....	42
4.2 Configuring OpenSSL .....	43

4.3 Testing Hardware Crypto functions . . . . .	45
4.3.1 Crypto Express6S card support for OpenSSL . . . . .	46
4.3.2 CPACF support for OpenSSL . . . . .	47
4.3.3 Testing the encryption using SSH. . . . .	49
<b>Chapter 5. IBM Secure Service Container . . . . .</b>	<b>51</b>
5.1 Introduction to IBM Secure Service Container . . . . .	52
5.2 IBM Secure Service Container internals . . . . .	52
5.2.1 Secure Service Container boot environment . . . . .	53
5.2.2 Boot sequence . . . . .	53
5.3 Installing and managing the appliance . . . . .	53
5.3.1 Installation and configuration . . . . .	53
5.3.2 Managing the appliance . . . . .	54
5.4 Key features of Secure Service Container . . . . .	55
<b>Related publications . . . . .</b>	<b>57</b>
IBM Redbooks . . . . .	57
Online resources . . . . .	57
Help from IBM . . . . .	57

# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of 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.

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

IBM®	Redpaper™	z/OS®
IBM LinuxONE™	Redpapers™	z/VM®
IBM z®	Redbooks (logo)  ®	z/VSE®
Redbooks®	System z®	z13®

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.



# Preface

This IBM® Redpaper™ publication discusses security practices for running Linux on IBM Z on the IBM z14.

It examines the unique security and integrity features that the IBM Z platform brings to the enterprise. It also examines pervasive encryption and its role in protecting data at rest.

## Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Lydia Parziale** is a Project Leader for the ITSO team in Poughkeepsie, New York, with international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include business development and database management technologies. Lydia is a certified PMP and an IBM Certified IT Specialist with an MBA in Technology Management and has been employed by IBM for over 25 years in various technology areas.

**Klaus Egeler** is an IT Systems Management Specialist with IBM GTS Delivery in Germany. He joined IBM in 1979. His areas of expertise are the mainframe operation systems IBM z/VSE®, IBM z/VM®, and Linux on IBM Z. Klaus has contributed to several z/VM and Linux-related IBM publications. He is a presenter and instructor at ITSO workshops and customer events around the world regularly.

**Manoj S. Pattabhiraman** is an Open Source enthusiast and an specialist from IBM ASEAN - LinuxONE Technical Sales Team. He has more than 16 years of experience in open source solutions (LinuxONE, Blockchain, Virtualization, and Cloud). In his current role, he drives and provides consultation on LinuxONE solutions across Asia Pacific. Manoj has contributed to several IBM publications and has been a frequent speaker at various technical conferences and workshops on Cloud, Blockchain, and Virtualization.

Thanks to the following people for their contributions to this project:

Robert Haimowitz  
International Technical Support Organization, Poughkeepsie Center

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:  
[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



# Why security and encryption

This chapter describes why security matters and how encryption helps to achieve a secure environment.

Security is probably the most important topic in the IT industry these days because numerous reports of data breaches are in the news that give the involved companies a bad reputation. That kind of bad reputation could cause a company to go out of business.

This chapter describes how encryption helps to provide a secure IT environment on the IBM Z platform. It includes the following sections:

- ▶ Why security matters
- ▶ Why use encryption
- ▶ Pervasive encryption with the IBM z14
- ▶ Benefits of Hardware Crypto
- ▶ Verification of installed LIC 3863 using the SE

## 1.1 Why security matters

Security is essential in many ways and became the most important topic among C-level executives. And it is not only about the security of the most valuable assets many companies have, their data, but also about avoiding bad publicity due to data breaches. Unfortunately, data breaches are in the news too often these days. The risk is high not only for smaller companies, but also for large enterprises, ISPs, and even global web companies. Security in information technology is a broad field and covers these topics:

- ▶ Authentication to ensure identity (certificates)
- ▶ Key Exchange to exchange cryptographic keys and perform handshaking
- ▶ Confidentiality to ensure that a message can only be read by the intended receiver (encryption)
- ▶ Integrity to ensure that a received message is still the original one and was not altered (hash/MAC)
- ▶ Nonrepudiation to ensure that a message really came from a certain sender (signature)

Every company that handles customer information or offers services through internet platforms must make sure that processed data is secured against all threats.

All precautions to prevent data leakage and to ensure system and data integrity must be taken. It is no longer sufficient to state that data processing is secure today. You also must offer proof to auditors and comply with regulations to establish trust in your services. Most importantly, you must prevent a loss of revenue and reputation due to security exposures.

Therefore, it is a preferred practice to establish the strongest security mechanisms at all levels of data processing, including the physical security of the machine rooms at your data center (controlling access to the facilities) and implementing appropriate access levels to applications, programs, data, archives, and so on. The principle of least privilege should be implemented at all levels.

## 1.2 Why use encryption

Data protection and security are business imperatives, and regulatory compliance is increasing in complexity. Extensive use of encryption is one of the best ways to reduce the risks and financial losses of a data breach and meet complex compliance mandates. However, implementing encryption can be a complex process for organizations. They need to determine these factors:

- ▶ What data should be encrypted?
- ▶ Where should encryption occur?
- ▶ Who is responsible for encryption?

Because the data is the new perimeter, encryption policies must cover both data in-flight and data at-rest, but should not require costly application changes to achieve this goal. Organizations need a transparent and consumable approach to enable extensive encryption of data in-flight and at-rest to substantially simplify and reduce the costs associated with protecting the data at the core of their enterprise and achieving compliance mandates.

With solutions around privileged identity management, sensitive data protection, and integrated security intelligence, IBM Z security offers the next generation of secure, trusted transactions.

## 1.3 Pervasive encryption with the IBM z14

Pervasive encryption is a data-centric approach to information security that entails protecting data entering and exiting the z14 platform. It involves encrypting data in-flight and at-rest to meet complex compliance mandates and reduce the risks and financial losses of a data breach. It is a paradigm shift from selective encryption (where only the data that is required to achieve compliance is encrypted) to pervasive encryption. Pervasive encryption with z14 is enabled through tight platform integration that includes these features:

- ▶ Integrated cryptographic hardware: Central Processor Assist for Cryptographic Function (CPACF) is a co-processor on every processor unit that accelerates encryption. Crypto Express features can be used as hardware security modules (HSMs).
- ▶ Data set and file encryption: You can protect Linux file systems and IBM z/OS® data sets by using policy-controlled encryption that is transparent to applications and databases.
- ▶ Network encryption: You can protect network data traffic by using standards-based encryption from endpoint to endpoint.
- ▶ Full disk encryption: You can use disk drive encryption that protects data at rest when disk drives are retired, sent for repair, or repurposed.
- ▶ CF encryption: Secures the parallel sysplex infrastructure including the CF links and data stored in the CF, using policy-based encryption.
- ▶ Secure Service Container: Secure deployment of software appliances including tamper protection during installation and run time, restricted administrator access, and encryption of data and code in-flight and at-rest.

Pervasive encryption has the following advantages:

- ▶ The ability to encrypt data by policy without application change.
- ▶ A simplified way to protect data at a much coarser scale with industry best performance.
- ▶ Greatly simplified audit, enabling clients to pass compliance audits more easily.

The z14 excels with security features that are built into the hardware, firmware, and operating systems. The built-in features range from storage protection keys and workload isolation to granular audit capabilities, and more. The CPACF, standard on every core, supports pervasive encryption and provides hardware acceleration for encryption operations. In addition, the new Crypto Express6S gets a performance boost on z14. Combined, these two enhancements perform encryption more efficiently on the z14 than on earlier Z platforms.

Security in individual layers might be enough to keep the data integrity, confidentiality, and availability at the destination. However, it is important to secure the data while it is in transit during communication.

Some solutions can be implemented at the client side, but the organization cannot rely on client-side only security. Users might forget to update their security software, security operating system updates might unknowingly install malware on their devices that prevents the execution of the security software, or the users might not install the security software.

What the organization can do is make sure the communication between the client and the server is encrypted with a secure cryptographic protocol. New vulnerabilities are often discovered on cryptographic protocols, cipher algorithms, and protocol implementation, so the security team must be up to date about what is secure to be used, and new vulnerabilities that must be mitigated as soon as they are reported.

Servers of the IBM Z family provide two different types of hardware support for cryptographic operations: CPACF and Crypto Express (CEX) features.

The z14 provide cryptographic functions that, from an application program perspective, can be grouped as follows:

- ▶ Synchronous cryptographic functions, provided by the CPACF or the Crypto Express features when defined as an accelerator.
- ▶ Asynchronous cryptographic functions, provided by the Crypto Express features.

### 1.3.1 CP Assist for Cryptographic Function

CPACF (shown in Figure 1-1) offers a set of symmetric cryptographic functions for high-performance encryption and decryption with clear key operations for SSL/TLS, VPN, and data-storing applications that do not require FIPS 140-2 level 4 security. CPACF is an optional feature that is integrated with the compression unit in the coprocessor in the z14 microprocessor core. CPACF is available on every Processor Unit defined as a CP, IFL, zAAP, and zIIP.

The CPACF protected key is a function that facilitates the continued privacy of cryptographic key material while keeping the wanted high performance. CPACF ensures that key material is not visible to applications or operating systems during encryption operations. A CPACF protected key provides substantial throughput improvements for large-volume data encryption and low latency for encryption of small blocks of data.

The cryptographic assist includes support for the following functions (the functions in bold are new with z14):

- ▶ Hashing algorithms (Hashes), message authentication codes (MACs): SHA1, SHA 2 (224,256, 384, 512), **SHA3 (224, 356, 384, 512)**, **SHAKE (128, 256)**, GHASH
- ▶ Symmetric ciphers: Data Encryption Standard (**DES, 2DES, 3DES**), Advanced Encryption Standard (AES-128, AES-192, AES-256)
- ▶ Modes of operations: ECB, CBC, CTR, OFB, CFB, XTS, CBC-MAC, **GCM**, (CMAC, CCM)
- ▶ Pseudo random number generation: 3DES based PRNG, NIST SP-800-90A SHA-512 based DRNG
- ▶ **True random number generation**
- ▶ Protected key support for additional security of cryptographic keys

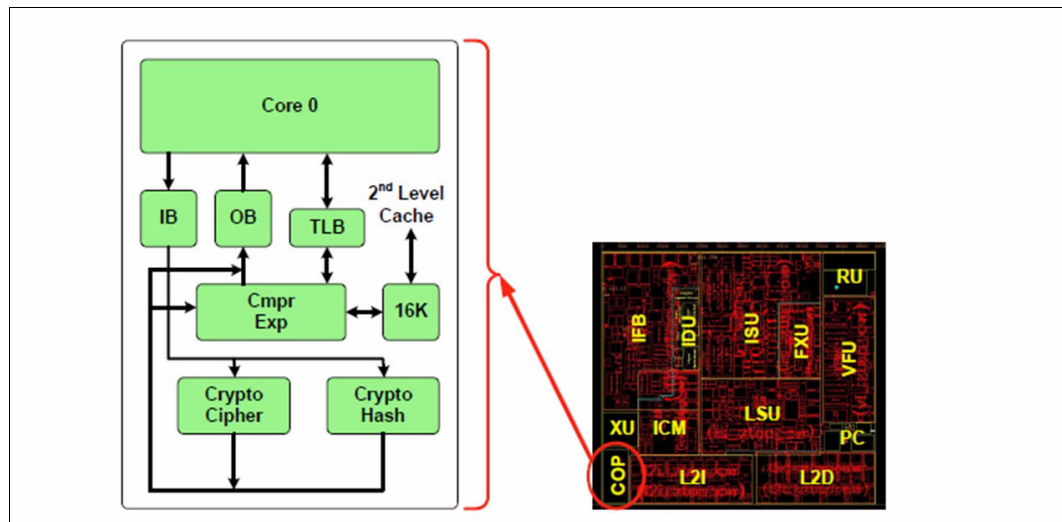


Figure 1-1 CPACF - CP Assist for Cryptographic Function (IBM z13@ schemes)

SHA-1, SHA-2, and SHA-3 support are enabled on all IBM Z platforms and do not require the CPACF enablement feature. The CPACF functions are supported by z/OS, z/VM, z/VSE, z/TPF, and Linux on IBM Z.

### 1.3.2 Crypto Express6S

The Crypto Express6S represents the newest generation of the Peripheral Component Interconnect Express (PCIe) cryptographic coprocessors, an optional feature exclusive to the z14. These are HSMs designed to provide high-security cryptographic processing as required by the banking and other industries. This feature provides a secure programming and hardware environment wherein crypto processes are performed.

Each cryptographic coprocessor includes general-purpose processors, non-volatile storage, and specialized cryptographic electronics, all contained within a tamper-sensing and tamper-responsive enclosure that destroys all keys and sensitive data on any attempt to tamper with the device. The security features of the HSM are designed to meet the requirements of FIPS 140-2, Level 4, the highest security level defined.

The Crypto Express6S has one PCIe adapter per feature. For availability reasons, a minimum of two features is required. Up to 16 Crypto Express6S features are supported. The Crypto Express6S feature occupies one I/O slot in a PCIe I/O drawer.

Each adapter can be configured as one of these objects:

- ▶ Secure IBM CCA coprocessor
- ▶ Secure IBM Enterprise PKCS #11 (EP11) coprocessor (since CEX4S)
- ▶ Accelerator

Crypto Express6S provides domain support for up to 85 logical partitions. Each domain has a separate master key.

Adapter management is done with the SE or the HMC by performing the following actions:

- ▶ Selection of adapter type (firmware load)
- ▶ Assignment of adapters and domains to LPARs

The accelerator function is designed for maximum-speed Secure Sockets Layer and Transport Layer Security (SSL/TLS) acceleration, rather than for specialized financial applications for secure, long-term storage of keys or secrets. The Crypto Express6S can also be configured as one of the following configurations:

- ▶ The Secure IBM CCA coprocessor includes secure key functions with emphasis on the specialized functions that are required for banking and payment card systems. It is optionally programmable to add custom functions and algorithms by using User Defined Extensions (UDX).

A new mode, called Payment Card Industry (PCI) PIN Transaction Security (PTS) Hardware Security Module (HSM), shortened to PCI-HSM, is available exclusively for Crypto Express6S in CCA mode. PCI-HSM mode simplifies compliance with PCI requirements for hardware security modules.

- ▶ The Secure IBM Enterprise PKCS #11 (EP11) coprocessor implements an industry-standardized set of services that adheres to the PKCS #11 specification v2.20 and more recent amendments. It was designed for extended FIPS and Common Criteria evaluations to meet industry requirements.

This cryptographic coprocessor mode introduced the PKCS #11 secure key function.

**TKE feature:** The Trusted Key Entry (TKE) Workstation feature is required for supporting the administration of the Crypto Express6S when configured as an Enterprise PKCS #11 coprocessor or managing the new CCA mode PCI-HSM.

When the Crypto Express6S PCI Express adapter is configured as a secure IBM CCA co-processor, it still provides accelerator functions. However, up to three times better performance for those functions can be achieved if the Crypto Express6S PCI Express adapter is configured as an accelerator.

## 1.4 Benefits of Hardware Crypto

The encryption of data is expensive and can heavily impact performance, throughput, or CPU load of a system. IBM Z provides hardware encryption support that can be used to reduce the impact of expensive encryption operations. Because the encryption operations are offloaded to the z14 CPACF processor or to the Crypto Express6S card, there is less impact on the performance and throughput of your workload.

IBM Z makes it possible, for the first time, for organizations to pervasively encrypt data associated with an entire application, cloud service, or database in flight or at rest with one click. The standard practice today is to encrypt small chunks of data at a time, and invest significant labor to select and manage individual fields.

This bulk encryption at cloud scale is made possible by a massive 7x increase in cryptographic performance over the previous generation z13. This increase is driven by a 4x increase in silicon dedicated to cryptographic algorithms. This rate is 18x faster compared to x86 systems (that today only focus on limited slices of data) and at just five percent of the cost of comparable x86-based solutions.

A top concern for organizations is protection of encryption keys. In large organizations, hackers often target encryption keys, which are routinely exposed in memory as they are used. Only IBM Z can protect millions of keys (and the process of accessing, generating, and recycling them) in “tamper responding” hardware that causes keys to be invalidated at any sign of intrusion. The keys can then be restored in safety.

The IBM Z key management system is designed to meet Federal Information Processing Standards (FIPS) Level 4 standards, whereas the norm for high security in the industry is Level 2. This IBM Z capability can be extended beyond the mainframe to other devices, such as storage systems and servers in the cloud. In addition, IBM Secure Service Container protects against insider threats from contractors and privileged users, provides automatic encryption of data and code in-flight and at-rest, and tamper-resistance during installation and run time.



Figure 1-2 shows an overview of Crypto support in IBM z14. You can find more information about this feature in an online [IBM news release](#).

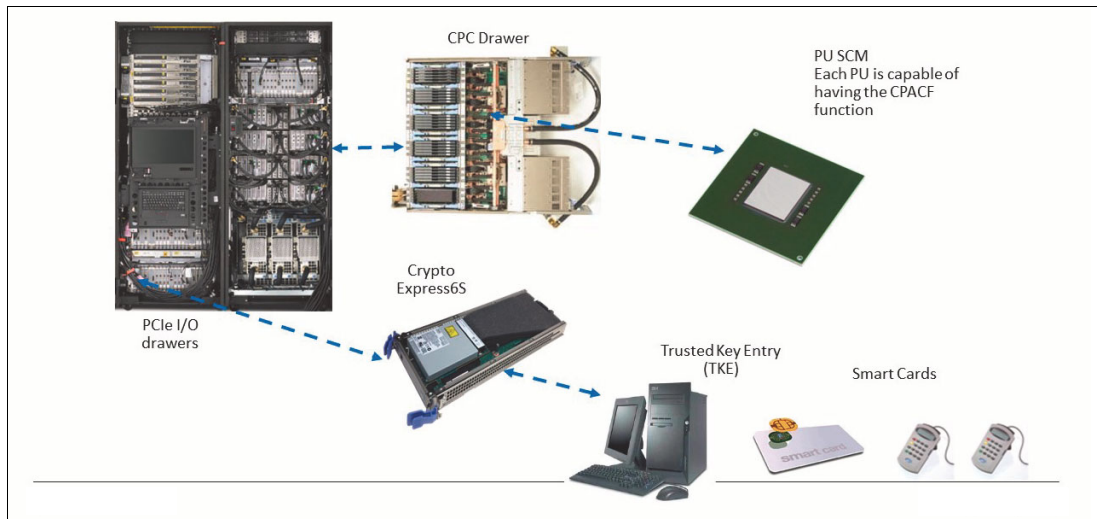


Figure 1-2 Overview of Hardware Crypto support in IBM z14

## 1.5 Verification of installed LIC 3863 using the SE

To benefit from the CPACF, you must install the Licensed Internal Code (LIC) feature 3863 (Crypto Enablement feature), which is available at no extra charge. The installation of this feature is a nondisruptive operation.

Install the Crypto Enablement feature even if you do not intend to use the Crypto Express feature because it provides considerable benefits to an active CPACF.

You can check whether CPACF is enabled in your environment by using the windows provided on the Support Element (SE).

Open the Hardware Management Console (HMC) web user interface in your browser and complete these steps:

1. Select **Tasks Index**.
2. Find **Single Object Operations**.
3. Switch to the Support Element (SE) by selecting **Single Object Operations**.
4. Select your IBM Z system and click **OK**.
5. Confirm that you want to establish a session by clicking **Yes**.
6. Select **Tasks Index** at the SE.
7. Find and select **System Details**.
8. Select your system and click **OK**.
9. Check for the phrase **CP Assist for Crypto functions: Installed** as shown in Figure 1-3 on page 8.

IBM Support Element

Home System Details - CETUS

CETUS Details - CETUS

Instance Information Product Information Acceptable CP/PCHID Status STP Information Energy Management Security

Group:

CP status: Operating

Channel status: Channel acceptable

Crypto status: Channel acceptable

Alternate SE status: Operating

Activation profile: DEFAULT

Last profile used: DEFAULT

IOCDS identifier: A0

IOCDS name: IODF40

System mode: Logically Partitioned

Service state: false

Number of CPs: 41

Number of ICFs: 0

Number of IFLs: 48

Number of zIIPs: 16

Dual AC power maintenance: Fully Redundant

CP Assist for Crypto functions: **Installed**

Primary Licensed Internal Code security mode: Monitor

Alternate Licensed Internal Code security mode: Monitor

Lock out disruptive tasks: ☐ Yes ☒ No

OK Apply Change Options... Cancel Help

Figure 1-3 IBM z14, LIC 3863 is installed



## Data security and Linux on IBM Z

This chapter describes ways that you can secure your data, your operating systems, and applications and overall infrastructure in the cloud. It also introduces some of the basic security concepts in the areas of networking, operating systems, data encryption, and operational controls.

This chapter includes the following sections:

- ▶ Data security
- ▶ Pervasive encryption
- ▶ LinuxONE and IBM Z Cryptographic Hardware features
- ▶ Overview of enabling cryptographic adapters
- ▶ Verification of installed LIC 3863 using the SE
- ▶ The hardware and software test environment
- ▶ Cryptographic software support: z/VM

## 2.1 Data security

The importance of data protection has become increasingly apparent with news reports of security breaches, loss and theft of personal and financial information, and government regulation. Securing the data at rest and data in flight helps control the risks of unauthorized data access without excessive security management burden.

Organizations need to consider factors like impact on performance, backup, security, and available resources to decide on proper encryption implementation. Businesses should consider the risks involved in losing the data that they handle, but also how long they need to keep data encrypted and how well they would be able to manage encrypting keys with each solution.

### 2.1.1 Encryption

Encryption is necessary, and is the best mechanism to protect data confidentiality, integrity, and genuineness. It minimizes the chance of security breaches and adds layers of protection to secure data. Costs related to data loss and requirements dictated by law should be incentive enough for all businesses to adopt solutions, regardless of whether they are hardware-based or software-based.

#### **Software encryption**

Software encryption is readily available for all major operating systems, and can protect data at rest, in transit, and stored on different devices. Software-based encryption often includes additional security features that complement encryption, which cannot come directly from the hardware.

##### ***Advantages***

Software encryption has these advantages:

- ▶ Pure software implementation
- ▶ Portable and supports arbitrary algorithms
- ▶ Additional security features are included for overall security (for example, to deal with email spam)

##### ***Disadvantages***

Software encryption has these disadvantages:

- ▶ Performance degradation and costs huge CPU cycles because the general processor would be stressed for cryptographic processing
- ▶ General Purpose CPUs are not optimized
- ▶ Limited to algorithms that are supported by the general-purpose CPU
- ▶ Less-than-perfect protection because hackers can exploit OS and memory vulnerabilities that expose encryption keys

#### **Hardware Encryption**

Hardware-based encryption uses speciality processor on-board to perform encryption and decryption. It is self-contained and does not require any additional software. Therefore, it is essentially free from the possibility of contamination, malicious code infection, or vulnerability. It requires simple or minimum configuration and user interaction, and does not cause much performance degradation during cryptographic processing.

A hardware-based solution is advisable when protecting sensitive data, and is also effective when protecting data at rest. Disks containing sensitive data pertaining to financial, healthcare, or government fields are better protected through hardware keys that can be effective even if drives are stolen and installed in other computers.

### **Advantages**

Hardware encryption has these advantages:

- ▶ Hardware-based encryption provides an integrated and layered security posture to achieve protection. It includes several ciphers and hashing algorithms.
- ▶ Separate crypto processor accelerates processing for better performance and integrated backup management.
- ▶ In the long run, hardware encryption can reduce the costs associated with IT labor, user productivity, and licensing fees.
- ▶ Provides tamper detection, secure key exchange, and authorization.

### **Disadvantages**

Hardware encryption has these disadvantages:

- ▶ Initial investment on speciality hardware for encryption is considered to be expensive.

As the need for computational complexity has increased, the time to encrypt and decrypt has also increased. Therefore, the desire for hardware acceleration has grown among organizations for security reasons.

## **2.2 Pervasive encryption**

Data protection and security are business imperatives, and regulatory compliance is increasing in complexity. Extensive use of encryption is one of the best ways to reduce the risks and financial losses of a data breach and meet complex compliance mandates. However, implementing encryption can be a complex process for organizations. They need to determine these factors:

- ▶ What data should be encrypted?
- ▶ Where should encryption occur?
- ▶ Who is responsible for encryption?

Because the data is the new perimeter, encryption policies must cover both data in-flight and data at-rest. However, they should not require costly application changes to achieve this goal. Organizations need a transparent and consumable approach to enable extensive encryption of data in-flight and at-rest to substantially simplify and reduce the costs associated with protecting the data at the core of their enterprise and achieving compliance mandates.

With solutions around privileged identity management, sensitive data protection, and integrated security intelligence, IBM Z security offers the next generation of secure, trusted transactions.

Pervasive encryption is a data-centric approach to information security that entails protecting data entering and exiting the z14 platform. It involves encrypting data in-flight and at-rest to meet complex compliance mandates and reduce the risks and financial losses of a data breach. It is a paradigm shift from selective encryption (where only the data that is required to achieve compliance is encrypted) to pervasive encryption.

Pervasive encryption with z14 is enabled through tight platform integration that includes these features:

- ▶ Integrated cryptographic hardware: CPACF is a co-processor on every processor unit that accelerates encryption. Crypto Express features can be used as hardware security modules (HSMs).
- ▶ Data set and file encryption: You can protect Linux file systems by using policy-controlled encryption that is transparent to applications and databases.
- ▶ Network encryption: You can protect network data traffic by using standards-based encryption from endpoint to endpoint.
- ▶ Full disk encryption: You can use disk drive encryption that protects data at rest when disk drives are retired, sent for repair, or repurposed.
- ▶ Secure Service Container: Secure deployment of software appliances including tamper protection during installation and run time, restricted administrator access, and encryption of data and code in-flight and at-rest.

Pervasive encryption has the following advantages:

- ▶ The ability to encrypt data by policy without application change
- ▶ A simplified way to protect data at a much coarser scale with industry best performance
- ▶ Greatly simplified audit, enabling clients to pass compliance audits more easily

## 2.3 LinuxONE and IBM Z Cryptographic Hardware features

The IBM Z and LinuxONE systems provide cryptographic functions that, from an application program perspective, can be grouped as follows (Figure 2-1):

- ▶ Synchronous cryptographic functions, provided by the CP Assist for Cryptographic Function (CPACF) or the Crypto Express features when defined as an accelerator.
- ▶ Asynchronous cryptographic functions, provided by the Crypto Express features.

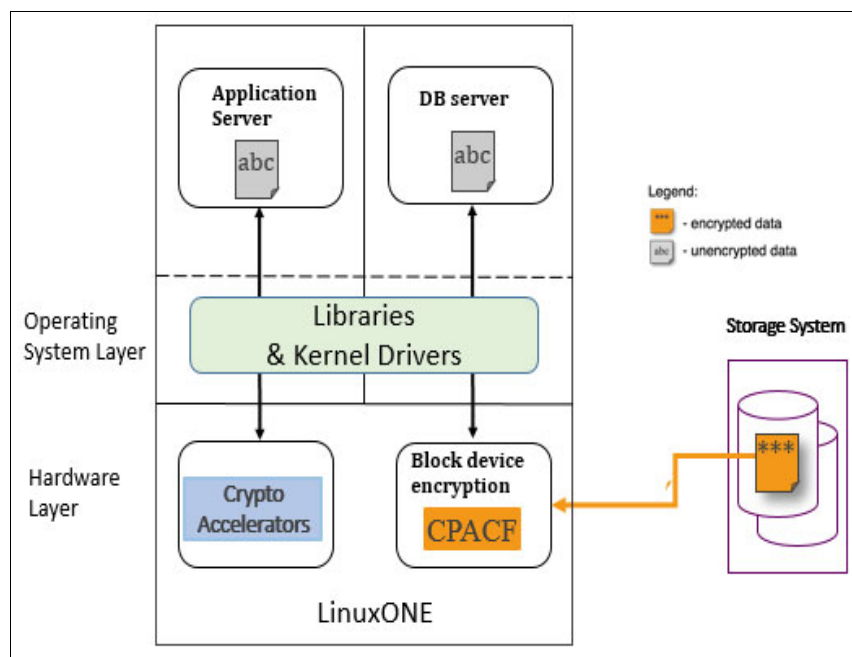


Figure 2-1 Cryptographic functions

### 2.3.1 CP Assist for Cryptographic Function

CP Assist for Cryptographic Function (CPACF) offers a set of symmetric cryptographic functions for high-performance encryption and decryption with clear key operations for SSL/TLS, VPN, and data-storing applications that do not require FIPS 140-2 level 4 security.

The CPACF is an optional feature that is integrated with the compression unit in the coprocessor in the z14 microprocessor core. The CPACF protected key is a function that facilitates the continued privacy of cryptographic key material while keeping the wanted high performance.

CPACF ensures that key material is not visible to applications or operating systems during encryption operations. CPACF protected key provides substantial throughput improvements for large-volume data encryption and low latency for encryption of small blocks of data.

The cryptographic assist includes support for the following functions:

- ▶ Advanced Encryption Standard (AES) for 128-bit, 192-bit, and 256-bit keys
- ▶ Data Encryption Standard (DES) data encryption and decryption with single, double, or triple length keys
- ▶ Pseudo-random number generation (PRNG)
- ▶ Message authentication code (MAC)
- ▶ Hashing algorithms: SHA-1, SHA-2, and SHA-3

SHA-1, SHA-2, and SHA-3 support are enabled on all IBM Z platforms and do not require the CPACF enablement feature. The CPACF functions are supported by z/OS, z/VM, z/VSE, z/TPF, and Linux on IBM Z.

### 2.3.2 Crypto Express6S

The Crypto Express6S represents the newest generation of the Peripheral Component Interconnect Express (PCIe) cryptographic coprocessors, an optional feature exclusive to the z14. These are hardware security modules (HSMs) designed to provide the high-security cryptographic processing required by banking and other industries. This feature provides a secure programming and hardware environment wherein crypto processes are performed.

Each cryptographic coprocessor includes general-purpose processors, non-volatile storage, and specialized cryptographic electronics, all contained within a tamper-sensing and tamper-responsive enclosure that destroys all keys and sensitive data on any attempt to tamper with the device. The security features of the HSM are designed to meet the requirements of FIPS 140-2, Level 4, the highest security level defined.

The Crypto Express6S has one PCIe adapter per feature. For availability reasons, a minimum of two features is required. Up to 16 Crypto Express6S features are supported. The Crypto Express6S feature occupies one I/O slot in a PCIe I/O drawer.

Each adapter can be configured as a Secure IBM CCA coprocessor, a Secure IBM Enterprise PKCS #11 (EP11) coprocessor, or as an accelerator. Crypto Express6S provides domain support for up to 85 logical partitions.

The accelerator function is designed for maximum-speed Secure Sockets Layer and Transport Layer Security (SSL/TLS) acceleration, rather than for specialized financial applications for secure, long-term storage of keys or secrets. The Crypto Express6S can also be configured as one of the following configurations:

- ▶ The Secure IBM CCA coprocessor includes secure key functions with emphasis on the specialized functions that are required for banking and payment card systems. It is optionally programmable to add custom functions and algorithms by using User Defined Extensions (UDX).

A new mode, called Payment Card Industry (PCI) PIN Transaction Security (PTS) Hardware Security Module (HSM), shortened to PCI-HSM, is available exclusively for Crypto Express6S in CCA mode. PCI-HSM mode simplifies compliance with PCI requirements for hardware security modules.

- ▶ The Secure IBM Enterprise PKCS #11 (EP11) coprocessor implements an industry-standardized set of services that adheres to the PKCS #11 specification v2.20 and more recent amendments. It was designed for extended FIPS and Common Criteria evaluations to meet industry requirements.
- ▶ This cryptographic coprocessor mode introduced the PKCS #11 secure key function.

**Note:** The Trusted Key Entry (TKE) Workstation feature is required for supporting the administration of the Crypto Express6S when configured as an Enterprise PKCS #11 coprocessor or managing the new CCA mode PCI-HSM.

When the Crypto Express6S PCI Express adapter is configured as a secure IBM CCA co-processor, it still provides accelerator functions. However, up to three times better performance for those functions can be achieved if the Crypto Express6S PCI Express adapter is configured as an accelerator.

## 2.4 Overview of enabling cryptographic adapters

This section broadly classifies the steps involved in enabling cryptographic environment for Linux on IBM Z and obtain the benefit of the additional power of special purpose features CPACF and IBM Crypto Express6SAccelerator (CEX6C). It also lists the high-level steps that are required to be followed for enabling and using Crypto features on Linux.

### 2.4.1 Hardware configuration: Setting up an LPAR to use Crypto

Complete these steps to set up an LPAR to use Crypto:

1. Using Support Element (SE), locate available crypto cards and domains.
2. After the Crypto domains and cards are identified, define them to an LPAR. Crypto definition can be done in following ways:
  - Statically: The definitions are permanently saved and require an LPAR deactivation and activation.
  - Dynamically: Crypto definitions are applied to an active LPAR, without any disruption.
3. Verification of installed Licensed Internal Code (LIC).
4. Define the cryptographic facility for the LPAR in which z/VM runs through the Hardware Configuration Definition and activate it.



## 2.4.2 Configuring Cryptographic adapters in z/VM

Complete these steps to configure Cryptographic adapters in z/VM:

1. Decide on whether the Crypto card is going to be dedicated or shared among the virtual machines.
2. Define the cryptographic capability for each Linux virtual machine in the user directory.
3. Using z/VM crypto commands, verify that the definitions have taken effect.

## 2.4.3 Setting up Cryptographic adapter for use from Linux on IBM Z

To set up the Cryptographic adapter for use, complete these steps:

1. Verify that the CPACF offloading capability is enabled in Linux on IBM Z. By default, this option is enabled.
2. Install the prerequisite device drivers and distribution RPM packages required for enabling Cryptographic processors (CEX6C).
3. Load Cryptographic device drivers into the kernel run time.
4. Verify that the algorithms are supported by hardware and software.

## 2.5 Verification of installed LIC 3863 using the SE

To benefit from the CPACF, you must install the LIC feature 3863 (Crypto Enablement feature), which is available at no extra charge. The installation of this feature is a nondisruptive operation.

Install the Crypto Enablement feature even if you do not intend to use the Crypto Express feature because it provides considerable benefits to an active CPACF.

You can check whether the CPACF is enabled in your environment by using the windows provided by the SE.

Open the Hardware Management Console (HMC) web user interface in your browser and complete these steps:

1. Select **Tasks Index**.
2. Find **Single Object Operations**.
3. Switch to the SE by selecting **Single Object Operations**.
4. Select your IBM Z system and click **OK**.
5. Confirm establishing a session by clicking **Yes**.
6. Select **Tasks Index** in the SE.
7. Find and select **System Details**.
8. Select your system and click **OK**.
9. Check for the phrase **CP Assist for Crypto functions: Installed** as shown in Figure 2-2 on page 16.

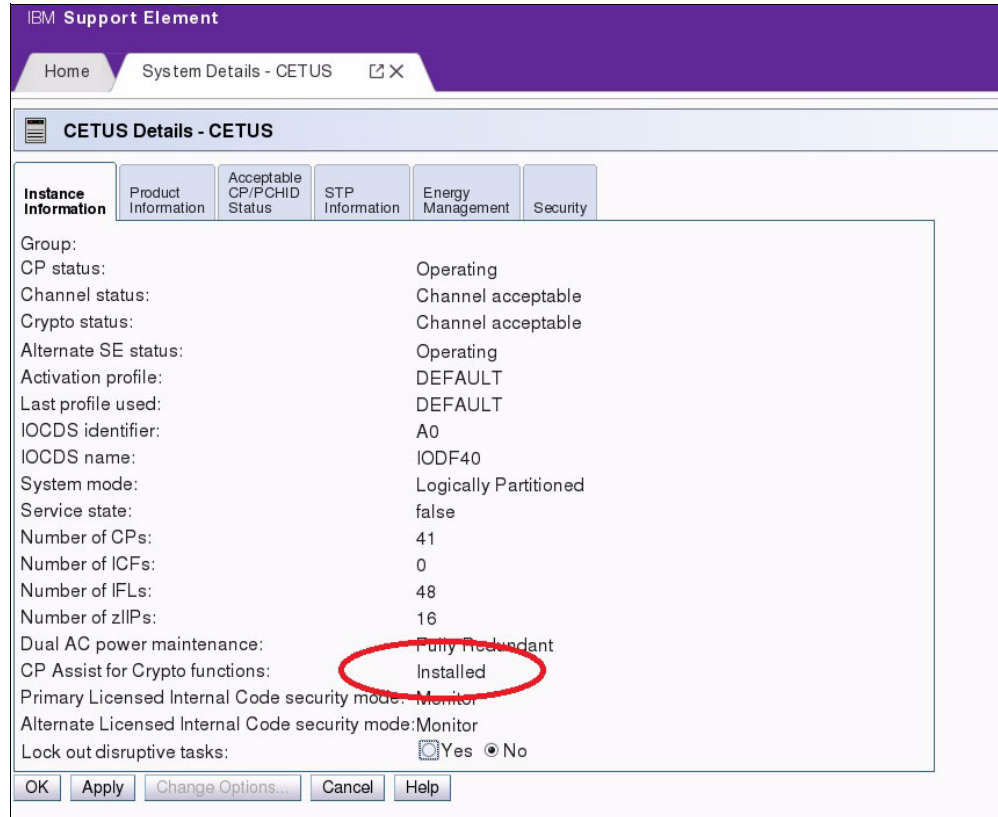


Figure 2-2 IBM z14, LIC 3863 is installed

## 2.6 The hardware and software test environment

For the test environment, two Crypto Express6S were enabled and configured for the z14 LPAR. Also, Red Hat Enterprise Linux servers were used as guests virtual machines on a z/VM v6.4 LPAR. z/VM 6.4 was updated with the latest patchset release (Example 2-1).

*Example 2-1 z/VM version and release information*

```
q cplevel
z/VM Version 6 Release 4.0, service level 1701 (64-bit)
Generated at 09/08/17 14:20:03 EDT
IPL at 09/08/17 16:25:44 EDT
Ready; T=0.01/0.01 13:58:26
```

We used the latest version of Red Hat Enterprise Linux release for the environment (Example 2-2), considering that the kernels provide toleration support for the Cryptographic 6S (CEX6S) adapters.

*Example 2-2 Linux kernel version*

```
[root@itsoln1 ~]# uname -na
Linux itsoln1 3.10.0-693.el7.s390x #1 SMP Thu Jul 6 20:01:53 EDT 2017 s390x s390x
s390x GNU/Linux
```

## 2.7 Cryptographic software support: z/VM

A z/VM guest virtual machine can either have one or more dedicated cryptographic devices or one shared cryptographic device, but not both:

- ▶ **Dedicated devices**

Each dedicated device maps to exactly one hardware device. The device representations in Linux on z/VM show the type of the actual hardware.

- ▶ **Shared device**

Virtual machines enable the sharing of IBM Z hardware among many operating systems. As a virtualization solution, the z/VM operating system does not provide a direct interface into any of the cryptographic hardware, nor does it require cryptographic services itself. z/VM provides a means of sharing the hardware cryptographic resources among the operating systems that are hosted (known as guests). Cryptographic resources can be shared through z/VM as follows:

- For all available cryptographic accelerators, z/VM provides unlimited access to all guests. This includes access to CPACF, the PCI Cryptographic Accelerator (PCICA), and the CEX6C when configured as a pure accelerator (CEX6A).
- z/VM makes Crypto Express available to guests with either dedicated access for use for both secure-key and clear-key operations, or with shared access for clear-key operations. Information on making Crypto Express available to a virtual machine can be found in the next section.

**Note:** No explicit z/VM authorization or configuration or definitions are required for accessing CPACF functions from z/VM.

The shared device can map to one or more hardware devices. The device representation in Linux on z/VM shows the type of the most advanced of these hardware devices. In this representation, cryptographic accelerators are considered more advanced than coprocessors.

### 2.7.1 Defining Cryptographic feature in z/VM

In z/VM, real cryptographic coprocessors are assigned to guest virtual machines by specifying the CRYPTO parameter in the VM guest machine directory with the DOMAIN and APDEDICATED or APVIRTUAL operands. See z/VM CP Planning and Administration, SC24-6083, for further explanation of these parameters.

APDEDICATED specifies the numbers of APs that the virtual machine can use for dedicated access to the AP cryptographic facility. The DO MAIN operand also must be specified to indicate the coprocessor domain to access in the APs. The AP queues that are equivalent to the domains that are specified by all of the DOMAIN operands are then assigned to the guest for each AP specified on the APDED operand on all CRYPTO statements.

The APs must be selected from the set in the PCI Cryptographic Online List on the Crypto Image Profile Page for the z/VM logical partition, or from the candidate list of APs that have been brought online to the z/VM logical partition. The DOMAINs must be part of the set selected in the Usage Domain Index list in the Crypto Image Profile Page for the Logical Partition.

APVIRTUAL tells the z/VM Control Program that this virtual machine can share access to the clear-key functions of only the AP cryptographic facility with other virtual machines. z/VM drives the requests to whatever coprocessor is available. The DOMAIN operand is not necessary when specifying the APVIRT operand because z/VM rejects all requests for services that would involve the use of secure keys.

**Note:** z/VM manages a pool of hardware cryptographic queues that are shared among all the virtual servers using the cryptographic facility. Even though the hardware queues are shared, the data remains isolated and is not vulnerable or exposed to other Linux images.

Figure 2-3 is a simple example of a z/VM guest machine setup. The z/VM system logical partition has access to APs 5 and 6 in coprocessor mode and AP 2 in accelerator mode. The directory of the z/OS VM guest machine has been set up to specify dedicated access to domain 1 of AP 5 and 6. The Linux VM guest machines are sharing access to whatever accelerator is available to the z/VM system. No setup is required to share the CPACF because the facility is available to whichever guest program is dispatched on the PU.

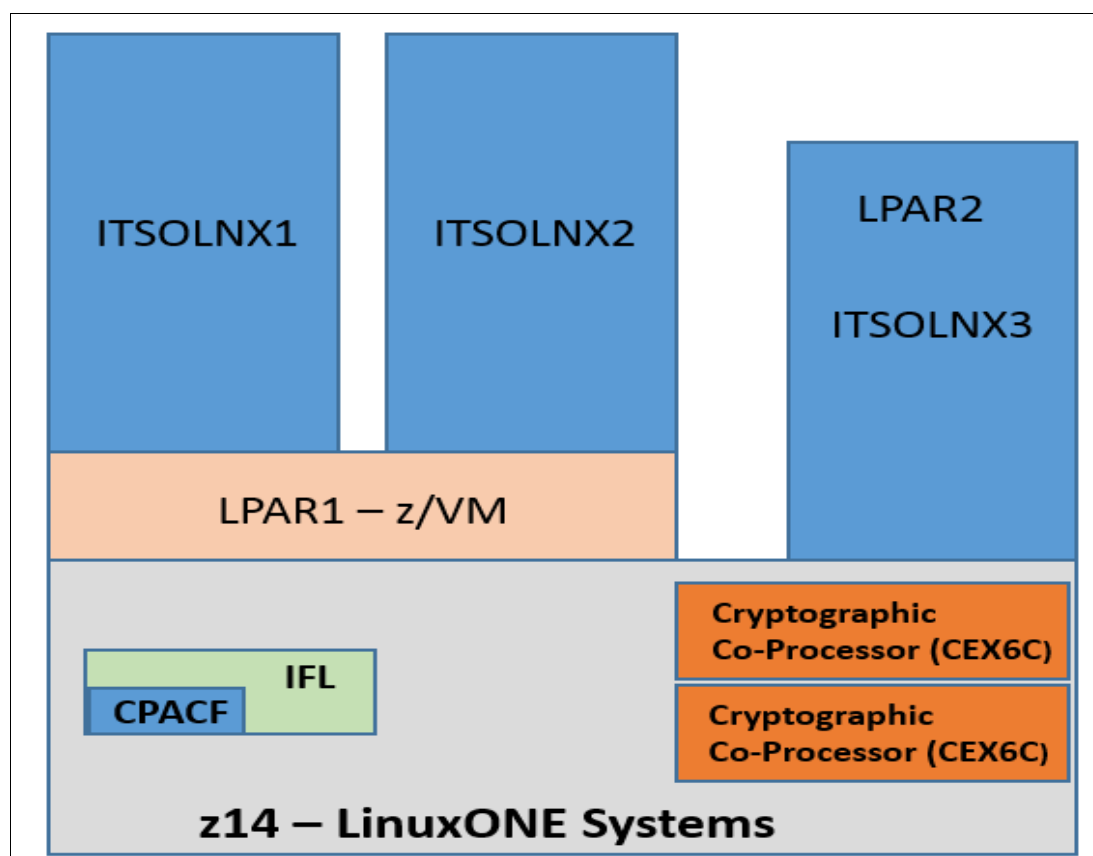


Figure 2-3 Hardware cryptographic coprocessors assignment to VM guest machines

## 2.7.2 Configuring Cryptographic feature in z/VM

When an LPAR has been configured to benefit from hardware cryptographic support, z/VM running in such an LPAR can use the hardware support for cryptographic operations to provide it to its guests. This section explains how to set up the z/VM definitions for guests running Linux on IBM Z.

The way z/VM provides this support is by gaining access to the Adjunct Processor (AP) queues to the guests. From a system implementation perspective, an AP of a CryptoExpress6s feature is one of its internal cryptography engines (cryptography coprocessor units). Note that AP designates to the processor, whereas AP ID specifies the number that is associated with it.

To use the accessible hardware by z/VM and to provide it to the guests, note the following rules:

- ▶ Each AP can have up to 16 usage domains assigned to it. Each usage domain has these characteristics:
  - Has a separate set of master keys for secure key operation stored in the CEX6C.
  - Is associated with a separate AP queue.
- ▶ The AP queues are in the hardware system area (HSA) and provide access to an AP.
- ▶ An AP queue can be identified by the AP number and the usage domain index.
- ▶ The AP numbers are assigned to the Cryptographic Candidate List or Cryptographic Online List in the LPAR activation profile.
- ▶ Each LPAR is assigned at least one usage domain that applies to all of the APs configured to this LPAR.
- ▶ The combination of usage domain and AP must be unique among active LPARs.

z/VM V6.3 and V6.4 provide the prerequisite IBM z14 encryption support to enable exploitation by guests in support of pervasive encryption of data in flight and at rest.

## z/VM Crypto Definition

The user directory statement CRYPTO APVIRT provides access to the cryptographic hardware and allows the z90crypt device driver to use cryptographic instructions. z/VM manages a pool of hardware cryptographic queues that are shared among all the virtual servers using the cryptographic facility. Example 2-3 shows the USER DIRECT statement with the crypto statement specifying shared cryptographic queues.

*Example 2-3 Sample USER DIRECT statement*

---

```

USER ITSOLNX1 XXXXXXXX 16G 16G BG
  IPL CMS
  MACHINE ESA 4
  CPU 00 NODEDICATE
  CRYPTO APVIRT
  CONSOLE 0009 3215 T
  NICDEF 0700 TYPE QDIO LAN SYSTEM VSWITCH1
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  LINK MAINT 0190 0190 RR
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
  MDISK 191 3390 0001 0100 IV1LX1 MR READ WRITE MULTI
  MDISK 200 3390 0101 END IV1LX1 MR READ WRITE MULTI

```

---

You can create more virtual servers that share the cryptographic facility than the actual number of hardware queues available. Even though the hardware queues are shared, the data remains isolated and is not vulnerable or exposed to other Linux images.

**Note:** z/VM does not have the capability to provide any cryptographic activity information, but it can show which VM guests are entitled to use hardware cryptographic coprocessors.

In case the system runs under z/VM, the availability of the crypto queue can be verified with the command shown in Example 2-4. The CP QUERY CRYPTO command can be used to query for the cryptographic coprocessor's physical configuration.

*Example 2-4 Verifying the availability of crypto queues*

---

```
q crypto
Crypto Adjunct Processor Instructions are installed
Ready; T=0.01/0.01 12:32:57

q crypto domain
AP 000 CEX6C Domain 012 available    shared          shared
AP 001 CEX6C Domain 012 available    shared          shared
Ready; T=0.01/0.01 12:33:03
```

---

This example of a QUERY Virtual CRYPTO command output shows that the z/VM system has access to two cryptographic coprocessors. Each coprocessor has domain 12 assigned to handle the requests coming from the logical partition where the z/VM instance resides.

After the co-processors are defined in the user directory, you can start configuring the crypto devices in Linux.

### 2.7.3 Hardware cryptography exploitation in Linux on IBM Z

Linux on z/VM with access to a shared cryptographic accelerator can either observe an accelerator or a coprocessor, but not both. When cryptographic coprocessors are shared, only clear-key RSA and random number functions are available to the Linux instance. Other requests are rejected by z/VM.

Figure 2-4 shows how the shared CEX6C card is defined in the z/VM Directory for the Linux virtual machine.

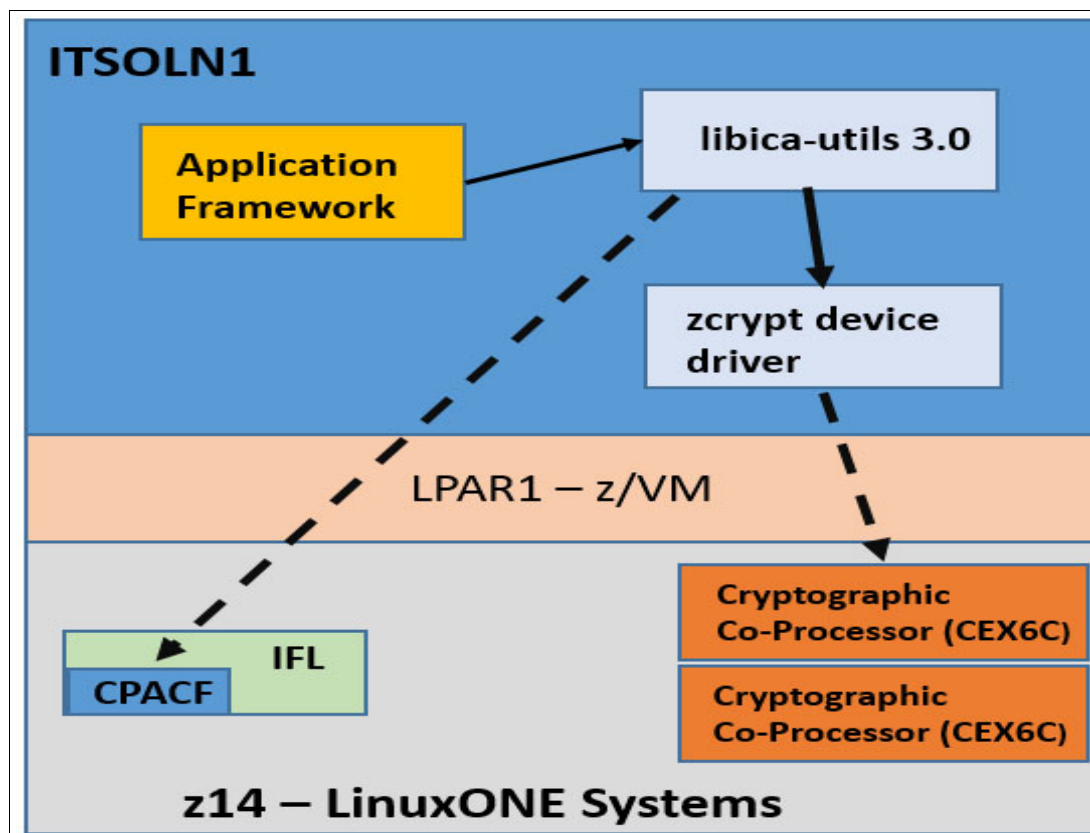


Figure 2-4 Linux on IBM Z device drivers and libraries

There are globally three layers of APIs. At the lowest level, the zcrypt, the device driver provides an API that is usually not used directly by applications. Instead, this API is intended for intermediate software layers, which in turn provide more sophisticated cryptographic functions to the next upper level of code.

The libica Linux library is an intermediate cryptographic functions library that offers a wide range of cryptographic functions. Some of these functions are performed by the hardware cryptographic devices under control of the device driver.

### CPACF Enablement verification

A Linux on IBM Z user can easily check whether the Crypto Enablement feature is installed and which algorithms are supported in hardware. Hardware-acceleration for DES, TDES, AES, and GHASH requires CPACF.

Read the features line from `/proc/cpuinfo` to discover whether the CPACF feature is enabled on your hardware as shown in Example 2-5.

*Example 2-5 Verify CPACF enabled in hardware*

```
[root@itsoln1 ~]# cat /proc/cpuinfo
vendor_id      : IBM/S390
# processors    : 1
bogomips per cpu: 21881.00
```

```
features      : esan3 zarch stfle msa ldisp eimm dfp edat etf3eh highprsr te vx
sie
cache0       : level=1 type=Data scope=Private size=128K line_size=256
processor 0: version = FF, identification = 27E0F7, machine = 3906
```

---

From the **cpuinfo** output, you can find the features that are enabled in the central processors. If the features list has **msa** listed, it means that CPACF is enabled.

**Important:** For Crypto Express 6s support, IBM is working with its Linux distribution partners to provide support through maintenance or future distribution releases. You can find more information at the [Linux on IBM Z](#) home page.

For the Linux virtual machine to gain access to the crypto card, you must load a specialized crypto device driver. By default, the device drivers that are required for Crypto processing are not loaded as shown in Example 2-6.

*Example 2-6 Command to determine whether device drivers are loaded*

```
[root@itsoln1 dev]# lszcrypt
lszcrypt: error - cryptographic device driver zcrypt is not loaded!
```

---

Most of the distributions include a generic kernel image for the specific platform. These device drivers for the generic kernel image are included as loadable kernel modules because statically compiling many drivers into one kernel causes the kernel image to be much larger. This kernel might be too large to boot on computers with limited memory.

**Note:** Make sure the **s390-tools** package is installed before using the **zcrypt** commands. Usually this package is installed by default in distributions, but if not you need to install it manually.

Use the **lsmod** command to check whether the crypto device driver module is already loaded. If the module is not loaded, use the **modprobe** command to load the device driver module. Example 2-7 shows that the Linux system is not yet loaded with the crypto device driver modules, so you must load it manually. The cryptographic device driver consists of multiple, separate modules. You can configure the cryptographic device driver through module parameters when you load the AP bus module.

*Example 2-7 Listing of existing device driver modules loaded*

```
[root@itsoln1 ~]# lsmod
Module                Size  Used by
nls_utf8              12586  1
isofs                 54891  1
loop                  28746  2
ip6t_rpfilter         12711  1
:
ccwgroup              19383  1 qeth
qdio                  71752  2 qeth,qeth_l2
dm_mirror             27404  0
dm_region_hash        20992  1 dm_mirror
dm_log                19301  2 dm_region_hash,dm_mirror
dm_mod                170712  8 dm_log,dm_mirror

[root@itsoln1 ~]# lsmod | grep ap
```

---



**Note:** When loading the device drivers using module parameters (`modprobe`, `insmod`), the device modules are not loaded automatically during the next reboot. Therefore, it is always advisable to update the initial ramdisk (`initrd`) to make the modules persistent across reboots.

It is possible to manually request the loading of a module with the `modprobe` or `insmod` command after the bootup process and make it permanently part of the system. The device driver is now loaded as separate modules, where the main module is called `ap`. However, there is an alias name `z90crypt` that links to the `ap` main module. You can also use the `lszcrypt` command to verify whether the Cryptographic co-processor driver is loaded already in your Linux system as shown in Example 2-8.

*Example 2-8 Checking whether the cryptographic co-processor driver is loaded*

---

```
[root@itsoln1 dev]# modprobe ap

[root@itsoln1 dev]# lsmod | grep ap
zcrypt_api          34633  2 zcrypt_cex4,zcrypt_msgtype6
ap                  40210  3 zcrypt_cex4,zcrypt_api,zcrypt_msgtype6
[root@itsoln1 dev]#
```

---

Check whether you have plugged in and enabled your IBM cryptographic adapter and validate your model and type configuration (accelerator or coprocessor). Use the `lszcrypt` command to retrieve basic status information (Example 2-9).

*Example 2-9 Retrieve basic Crypto adapter information*

---

```
[root@itsoln1 ~]# lszcrypt
card01: CEX5C
[root@itsoln1 ~]#
```

---

## Installing libica 3.0

To make use of the libica hardware support for cryptographic functions, you must install the libica version 3.0 package. Obtain the current libica version 3.0 as an RPM package from your distribution provider for automated installation. Depending on the distribution and installation parameters, some or all of them might be already installed with your initial setup (Example 2-10).

*Example 2-10 Install the libica version 3.0 package*

---

```
[root@itsoln1 ~]# yum install libica-utils
Loaded plugins: product-id, search-disabled-repos, subscription-manager
This system is not registered with an entitlement server. You can use
subscription-manager to register.
Resolving Dependencies
--> Running transaction check
---> Package libica.s390x 0:3.0.2-2.e17 will be installed
--> Finished Dependency Resolution

:
:
Running transaction
Installing : libica-3.0.2-2.e17.s390x 1/1
repository/productid | 1.6 kB  00:00:00
Verifying : libica-3.0.2-2.e17.s390x 1/1
```

---

Installed:  
**libica.s390x 0:3.0.2-2.e17**

Complete!

**Note:** The **icainfo** and **icastats** commands are provided with the libica package and described in libica Programmer's Reference, SC34-2602. Read about the dependencies on software and hardware that exist if you want to run libica in Federal Information Processing Standard (FIPS) mode.

Use the **rpm** command to verify that the package is installed successfully and correctly as shown in Example 2-11.

*Example 2-11 Package installation status*

```
[root@itsoln1 dev]# rpm -qa | grep libica
libica-3.0.2-2.e17.s390x
```

After the libica utility is installed, use the **icainfo** command to check on the CPACF feature code enablement. If the Crypto Enablement feature 3863 is installed, you will see that besides SHA, other algorithms are available with hardware support.

The **icainfo** command displays which CPACF functions are supported by the implementation inside the libica library. The results in Example 2-12 command show that the device driver is not loaded and all the RSA-based requests would be handled by software implementation.

*Example 2-12 Verification of crypto algorithms supported by hardware and software*

```
[root@itsoln1 ~]# icainfo
Cryptographic algorithm support
-----
function      | hardware | software
-----+-----+-----
SHA-1         | yes      | yes
SHA-224       | yes      | yes
SHA-256       | yes      | yes
SHA-384       | yes      | yes
SHA-512       | yes      | yes
GHASH         | yes      | no
P_RNG         | yes      | yes
DRBG-SHA-512  | yes      | yes
RSA ME        | yes      | yes
RSA CRT       | yes      | yes
DES ECB       | yes      | yes
DES CBC       | yes      | yes
DES OFB       | yes      | no
DES CFB       | yes      | no
DES CTR       | yes      | no
DES CMAC      | yes      | no
3DES ECB      | yes      | yes
3DES CBC      | yes      | yes
3DES OFB      | yes      | no
3DES CFB      | yes      | no
3DES CTR      | yes      | no
3DES CMAC     | yes      | no
AES ECB       | yes      | yes
```

AES CBC	yes	yes
AES OFB	yes	no
AES CFB	yes	no
AES CTR	yes	no
AES CMAC	yes	no
AES XTS	yes	no

---

Built-in FIPS support: FIPS mode inactive.

---

Starting with libica version 3.0, the library is enabled for FIPS 140-2 certification and therefore can run in the so-called FIPS mode. When running in FIPS mode, only cryptographic algorithms approved by the National Institute of Standards and Technology (NIST) can be used.

## Enabling libica for FIPS Mode

To use libica in FIPS mode, the library itself and also the Linux kernel need to be enabled. That is, the FIPS-enabled libica library can run in FIPS mode when the kernel FIPS flag is set.

A prerequisite to running in the FIPS-enabled libica in FIPS mode is to set the FIPS flag in the used Linux kernel configured for FIPS. For all distributions, you need to enable the kernel FIPS mode at run time by setting the kernel FIPS flag. To set this flag in `/proc/sys/crypto/fips_enabled`, boot or reboot with the kernel parameter `fips=1`.

To determine what kernel parameters were used during your system's last boot, use the command shown in Example 2-13.

*Example 2-13 Kernel parameters in `zipl.conf`*

---

```
parameters="root=/dev/mapper/rhel_itsoln1-root crashkernel=auto rd.dasd=0.0.0200
rd.lvm.lv=rhel_itsoln1/root rd.lvm.lv=rhel_itsoln1/swap cio_ignore=all,!condev
rd.znet=qeth,0.0.0700,0.0.0701,0.0.0702,layer2=1,portname=foobar LANG=en_US.UTF-8"
```

---

Kernel parameters that you specify when booting Linux are not persistent. To define a permanent set of kernel parameters for a Linux instance, include these parameters in the boot configuration. There are multiple ways to enable kernel parameters. In this paper, we include the kernel parameter part of the boot configuration to make it persistent. For that, we edited `zipl.conf` and included `fips=1` in to the file as shown in Example 2-14.

*Example 2-14 Updated `zipl.conf` with `fips` kernel parameter*

---

```
[root@itsoln1 dev]# cat /proc/cmdline
root=/dev/mapper/rhel_itsoln1-root crashkernel=auto rd.dasd=0.0.0200
rd.lvm.lv=rhel_itsoln1/root rd.lvm.lv=rhel_itsoln1/swap cio_ignore=all,!condev
rd.znet=qeth,0.0.0700,0.0.0701,0.0.0702,layer2=1,portname=foobar LANG=en_US.UTF-8
fips=1 BOOT_IMAGE=0
```

---

After we making the changes to the `zipl.conf` file, use the `zipl` tool to create Linux boot configurations for Linux on IBM Z or LinuxONE systems, and then reboot so the inclusion takes effect (Example 2-15).

*Example 2-15 Create the Linux boot configurations*

---

```
[root@itsoln1 etc]# zipl
Using config file '/etc/zipl.conf'
Building bootmap in '/boot'
Building menu 'zipl-automatic-menu'
```

---

```

Adding #1: IPL section 'linux' (default)
Adding #2: IPL section 'linux-0-rescue-2d710d836f974bd3ae0d78a0db4c6699'
Preparing boot device: dasda (0200).
Done.
[root@itsoln1 etc]#

```

---

After the Linux system is rebooted, verify that the fips mode is activated and included part of the runtime kernel as shown in Example 2-16.

*Example 2-16 Verifying the kernel parameter is included part of the system.*

```

[root@itsoln1 ~]# cat /proc/cmdline
root=/dev/mapper/rhel_itsoln1-root crashkernel=auto rd.dasd=0.0.0200
rd.lvm.lv=rhel_itsoln1/root rd.lvm.lv=rhel_itsoln1/swap cio_ignore=all,!condev
rd.znet=qeth,0.0.0700,0.0.0701,0.0.0702,layer2=1,portname=foobar LANG=en_US.UTF-8
fips=1 BOOT_IMAGE=0

```

---

The **icainfo** output now indicates whether libica has built-in FIPS support, whether it is running in FIPS mode, and whether the algorithms that are not FIPS approved are marked as blocked when running in FIPS mode (Example 2-17).

*Example 2-17 Cryptographic adapters processing status*

```

[root@itsoln1 dev]# icainfo
Cryptographic algorithm support
-----

```

function	hardware	software
SHA-1	yes	yes
SHA-224	yes	yes
SHA-256	yes	yes
SHA-384	yes	yes
SHA-512	yes	yes
GHASH	yes	no
P_RNG	blocked	blocked
DRBG-SHA-512	yes	yes
RSA ME	yes	yes
RSA CRT	yes	yes
DES ECB	blocked	blocked
DES CBC	blocked	blocked
DES OFB	blocked	blocked
DES CFB	blocked	blocked
DES CTR	blocked	blocked
DES CMAC	blocked	blocked
3DES ECB	yes	yes
3DES CBC	yes	yes
3DES OFB	yes	no
3DES CFB	yes	no
3DES CTR	yes	no
3DES CMAC	yes	no
AES ECB	yes	yes
AES CBC	yes	yes
AES OFB	yes	no
AES CFB	yes	no
AES CTR	yes	no
AES CMAC	yes	no

AES XTS |      yes      |      no     

-----  
Built-in FIPS support: FIPS mode active.  
[root@itsoln1 dev]#

---

**Important:** FIPS mode needs to be enabled or disabled based on your regulatory and organizational requirements.

For the purpose of this paper, we disabled the fips mode and continued with our configuration and use cases.





## Pervasive encryption: Data-at-rest encryption

This chapter describes how to pervasively encrypt data-at-rest without changes to applications or performance by using the inherent capabilities of LinuxONE systems. Using cryptographic capabilities of IBM Z, you can speed up the process of protecting file system data using encryption.

This chapter includes the following sections:

- ▶ Introduction
- ▶ LinuxONE Hardware-accelerated in-kernel cryptography
- ▶ Data-at-Rest using dm-crypt LUKS encryption

## 3.1 Introduction

With LinuxONE, it is now possible for organizations to pervasively encrypt data associated with an entire application, cloud service, or database in flight or at rest with one click. The standard practice today is to encrypt small chunks of data at a time, and invest significant labor to select and manage individual fields. This bulk encryption at cloud scale is made possible by a massive 7x increase in cryptographic performance and also driven by a 4x increase in silicon dedicated to cryptographic algorithms.

The following are the main ways to encrypt your data on disk with Linux:

- ▶ Implementing data encryption in the application
- ▶ Using a file system that encrypts only the file contents on disk
- ▶ Using a file system with built-in encryption
- ▶ Encrypting a block device and using a regular file system on that block device

Even though each of the options has its own pros and cons, this section looks at transparently encrypting whole block devices and file systems by users of in-kernel crypto in Linux.

## 3.2 LinuxONE Hardware-accelerated in-kernel cryptography

Linux provides encryption support for these areas:

- ▶ In-kernel cryptography
- ▶ Cryptographic support for user programs or applications

In-kernel cryptography is used when the Linux kernel itself performs encryption requests by using in-kernel modules. Because the in-kernel modules are not available for user programs and applications, there are specific encryption libraries available.

To use hardware support for encryption, the in-kernel cryptography modules or the cryptography libraries must be aware of the available hardware and be able to use it.

### 3.2.1 Verification of support for Hardware Cryptographic operation

To find the operations that can be performed using hardware cryptography, issue the command shown in Example 3-1. Each line that ends in `-s390` indicates hardware-acceleration for a corresponding algorithm or mode.

*Example 3-1 Supported hardware Cryptographic operations*

---

```
[root@itsoln1 crypto]# cat /proc/crypto | grep driver
driver      : crc32c-generic
driver      : aes-generic
driver      : sha224-generic
driver      : sha256-generic
driver      : sha1-generic
driver      : md5-generic
driver      : sha224-s390
driver      : sha256-s390
driver      : sha1-s390
```

---



Figure 3-1 shows a diagram of the configuration.

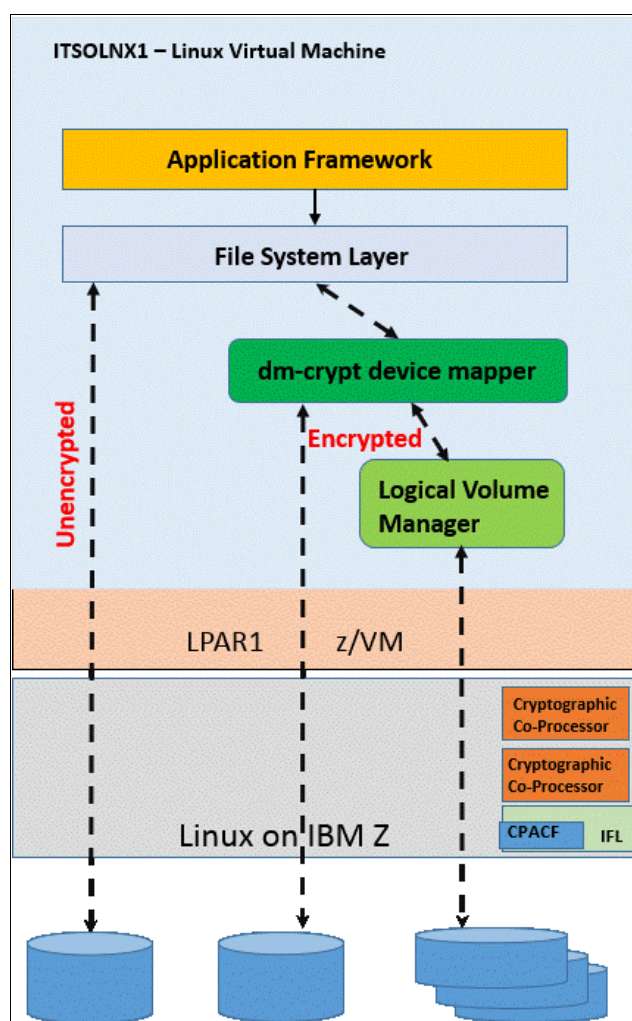


Figure 3-1 In-kernel cryptography working with the hardware

The IBM Z specific modules and libraries can autodetect which hardware support is available. The in-kernel modules are included with the kernel, whereas the support for user programs might be installed separately using Operating systems installers. On LinuxONE Systems, hardware-accelerated processing is available for some of these operations.

**Note:** For more information about hardware-accelerated in-kernel cryptography, see [IBM Knowledge Center](#).

### 3.2.2 The dm-crypt module

The Linux kernel implements cryptographic operations for kernel subsystems like dm-crypt and IPSec. The dm-crypt subsystem in Linux is implemented as a device mapper that can be stacked on the top of other devices that are managed through the device mapper framework. Therefore, you can encrypt from entire disks to software RAID volumes and LVM logical volumes, adding flexibility to your encryption strategy.

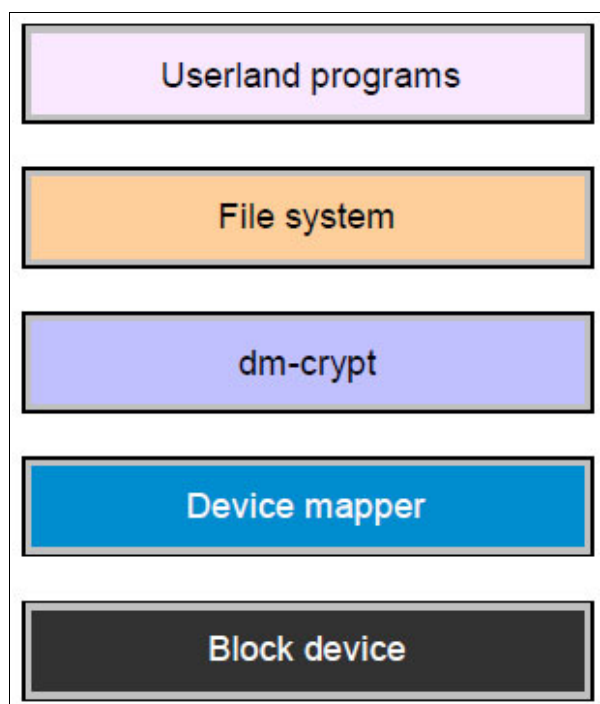
The cryptographic operations that can be accelerated by hardware implementations depend on your IBM Z hardware features and mode of operating Red Hat Enterprise Linux 7.4. z196

and later IBM Z hardware supports hardware-acceleration for operations that cover the following standards and more:

- ▶ SHA-1
- ▶ SHA-256
- ▶ SHA-512
- ▶ DES and TDES (ECB, CBC, and CTR modes)
- ▶ AES (ECB, CBC, and CTR modes for all AES key sizes; XTS for 256-bit and 512-bit keys)
- ▶ GHASH

One of the cryptography features of the Linux kernel is dm-crypt, which is a device mapper and a transparent disk encryption subsystem that allows you to encrypt whole block devices. The dm-crypt feature inserts a cryptographic layer between the block device and the accessing file systems or applications. You can use an encrypted block device like any other block device and install arbitrary file systems on it or use it as swap space.

Figure 3-2 shows the dm-crypt placed between the userland programs (user space and operating system software that does not belong in the kernel) and file systems and the device mapper and block device. With block device layer encryption, such as that with dm-crypt, the user creates the file system on the block device, and the encryption layer transparently encrypts the data before writing it to the actual lower block device. All file system operations from the userland pass through to dm-crypt.



*Figure 3-2 Block device layer encryption using dm-crypt*

The administration of dm-crypt is done using cryptsetup, which features Linux Unified Key Setup (LUKS). LUKS standardizes the format of the encrypted disk, which allows different implementations, even from other operating systems, to access and decrypt the disk.

LUKS adds metadata to the underlying block device, which contains information about the ciphers used and a default of eight key slots that hold an encrypted version of the master key used to decrypt the device. You can unlock the key slots by either providing a password on the command line or using a key file, which could, for example, be encrypted with gpg and stored on an NFS share.

The dm-crypt module supports various cipher and hashing algorithms that you can choose from the ones that are available in the Kernel and listed in the `/proc/crypto` `procfs` file. This feature also means that dm-crypt takes advantage of the unique hardware acceleration features of IBM Z that increase encryption and decryption speed.

### 3.3 Data-at-Rest using dm-crypt LUKS encryption

To showcase transparent full disk encryption, we attached a couple of disks to the virtual machine as shown in Example 3-2. The plan is to test transparent full disk encryption with HW assistance (CPACF) and without HW assistance.

*Example 3-2 Disk definition on z/VM*

---

```
[root@itsoln1 ~]# lsdasd
```

Bus-ID	Status	Name	Device	Type	BlkSz	Size	Blocks
0.0.0200	active	dasda	94:0	ECKD	4096	21058MB	5391000
0.0.0300	n/f	dasdb	94:4	ECKD			
0.0.0400	n/f	dasdc	94:8	ECKD			

---

```
[root@itsoln1 ~]#
```

**Note:** Make sure that the newly added DASD volumes are recognized during boot by whitelisting the devices as part of `cio_ignore` kernel parameter in `zipl.conf`.

Standard Linux kernel includes modules that use the CPACF capabilities of the IBM Z hardware. These modules are optimized to use the inherent IBM Z hardware features to provide better performance and security.

The hardware-dependent modules for the CPACF are typically located in the `/lib/modules` directory unless they have been compiled into the kernel itself. Example 3-3 shows making sure these modules have been included.

*Example 3-3 Verifying hardware dependent modules*

---

```
[root@itsoln1 crypto]# pwd
/usr/lib/modules/3.10.0-693.el7.s390x/kernel/arch/s390/crypto
```

```
[root@itsoln1 crypto]# ls -la
```

```
total 208
drwxr-xr-x. 2 root root 102 Sep 21 15:09 .
drwxr-xr-x. 7 root root 73 Sep 21 15:09 ..
-rw-r--r--. 1 root root 48313 Jul 6 20:58 aes_s390.ko
-rw-r--r--. 1 root root 42033 Jul 6 20:58 des_s390.ko
-rw-r--r--. 1 root root 28217 Jul 6 20:58 ghash_s390.ko
-rw-r--r--. 1 root root 55601 Jul 6 20:58 prng.ko
-rw-r--r--. 1 root root 28785 Jul 6 20:58 sha512_s390.ko
```

---

```
[root@itsoln1 crypto]#
```

To make sure that the CPACF hardware is used for the encryption, load the IBM Z optimized `aes_s390` kernel module, as in Example 3-4.

*Example 3-4 Loading the `aes_s390` module*

---

```
[root@itsoln1 ~]# modprobe aes_s390

[root@itsoln1 ~]# lsmod | grep aes_s390
aes_s390                18044  0
[root@itsoln1 ~]#

[root@itsoln1 ~]# cat /proc/crypto | grep aes-s390
driver      : xts(aes-s390)
driver      : xts-aes-s390
driver      : ctr-aes-s390
driver      : xts-aes-s390
driver      : cbc-aes-s390
driver      : ecb(aes-s390)
driver      : ecb-aes-s390
driver      : aes-s390
[root@itsoln1 ~]#
```

---

### 3.3.1 Setting up cryptsetup

The `cryptsetup` feature provides an interface for configuring encryption on block devices (such as `/home` or swap partitions), using the Linux kernel device mapper target `dm-crypt`. It features integrated LUKS support.

LUKS standardizes the format of the encrypted disk, which allows different implementations, even from other operating systems, to access and decrypt the disk. LUKS adds metadata to the underlying block device, which contains information about the ciphers used and a default of eight key slots that hold an encrypted version of the master key used to decrypt the device.

You can unlock the key slots by either providing a password on the command line or using a key file, which could, for example, be encrypted with `gpg` and stored on an NFS share.

Example 3-5 shows how to install the `cryptsetup` package.

*Example 3-5 Install `cryptsetup` package*

---

```
[root@itsoln1 iso]# yum install cryptsetup
Loaded plugins: product-id, search-disabled-repos, subscription-manager
This system is not registered with an entitlement server. You can use
subscription-manager to register.
repository
| 4.1 kB 00:00:00
Resolving Dependencies
:
:
Installed:
cryptsetup.s390x 0:1.7.4-3.el7

Complete!

[root@itsoln1 iso]# rpm -qa | grep crypt*
cryptsetup-libs-1.7.4-3.el7.s390x
```

```
cryptsetup-1.7.4-3.el7.s390x
libgcrypt-1.5.3-14.el7.s390x
m2crypto-0.21.1-17.el7.s390x
[root@itsoln1 iso]#
```

---

### 3.3.2 Creating a LUKS partition

The dm-crypt feature supports various cipher and hashing algorithms that you can select from the ones that are available in the Kernel and listed in the `/proc/crypto` procfs file. This also means that dm-crypt takes advantage of the unique hardware acceleration features of IBM Z that increase encryption and decryption speed.

Using the **cryptsetup** command, create a LUKS partition on the respective disk devices. For full disk encryption, use the AES-xts hardware feature. We choose the AES-xts to achieve a security level of reasonable quality with the best encryption mode.

**Important:** If the partition to be encrypted is going to be boot partition, when you choose to protect the device with a password, you must enter it interactively during the boot process on the console. The boot process does not continue until the password has been entered.

During the partition creation, you need to key in a passphrase. The encryption key is derived from this passphrase. After the user passphrase is derived, it is encrypted with the master key to produce the encrypted master key. This process is shown in Example 3-6.

*Example 3-6 Creating LUKS partition*

---

```
[root@itsoln1 iso]# cryptsetup luksFormat -c aes-xts-plain64:sha512 -s 512
/dev/dasdb1
```

```
WARNING!
```

```
=====
```

```
This will overwrite data on /dev/dasdb1 irrevocably.
```

```
Are you sure? (Type uppercase yes): YES
```

```
Enter passphrase:
```

```
Verify passphrase:
```

```
[root@itsoln1 iso]#
```

---

Now that the LUKS partition is created, you can dump the partition header to gather more information, as shown in Example 3-7.

*Example 3-7 LUKS partition dump*

---

```
[root@itsoln1 iso]# cryptsetup luksDump /dev/dasdb1
LUKS header information for /dev/dasdb1
```

```
Version:          1
Cipher name:      aes
Cipher mode:      xts-plain64:sha512
Hash spec:        sha256
Payload offset:   4096
MK bits:          512
MK digest:        27 c4 3f 6f b3 b6 d4 ed 1d 26 65 f5 3f 92 ec e8 01 3d a3 a2
MK salt:          33 c9 94 86 86 1c 7c e8 5e 4c 36 a6 1f 4d 6b ef
```

```

60 c1 4c 73 ee 95 f9 da c4 ed 1f 08 52 80 f2 57
MK iterations: 34750
UUID:         04d264d3-4759-4a14-96bc-76373f14e827

Key Slot 0: ENABLED
    Iterations:      281318
    Salt:            67 64 a1 5a a7 af 09 af 3b 78 55 af d1 7e f0 8f
                    80 02 d1 2c 93 ad 50 82 57 b4 e4 96 16 74 b3 94
    Key material offset: 8
    AF stripes:      4000
Key Slot 1: DISABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
[root@itsoln1 iso]#

```

---

The partition header dump in Example 3-7 on page 35 allows you to verify the following information about the master key (MK):

- Cipher: AES-256 with XTS mode
- Hash function: SHA256
- Master key length: 256 bits
- Master key digest: 27 c4 3f 6f b3 b6 d4 ed 1d 26 65 f5 3f 92 ec e8 01 3d a3 a2
- Master key salt: 567 64 a1 5a a7 af 09 af 3b 78 55 af d1 7e f0 8f 80 02 d1 2c 93 ad 50 82 57 b4 e4 96 16 74 b3 94
- Iteration for the master Key hash: 281318

At the moment, only one passphrase is enabled for this partition (key slot 0), but you can add more passphrases.

## Mapping and accessing the device partition

Now that you have some understanding on the partition header, create a mapping under `/dev/mapper`. As the device is encrypted, key in the passphrase to create the mapping as shown in Example 3-8.

### *Example 3-8 Creating mapping for the device*

```

[root@itsoln1 iso]# cryptsetup open --type luks /dev/dasdb1 myCryptodev
Enter passphrase for /dev/dasdb1:
[root@itsoln1 iso]#

```

---

The unlocking or mapping process maps the partitions to a new device name by using the device mapper. This process alerts the kernel that device is an encrypted device and should be addressed through LUKS using the `/dev/mapper/dm_name` so as not to overwrite the encrypted data (Example 3-9).

### *Example 3-9 Creation of mapping under device mapper*

```

[root@itsoln1 iso]# ll /dev/mapper/
total 0
crw----- 1 root root 10, 236 Sep 26 09:50 control

```

```
lrwxrwxrwx. 1 root root      7 Sep 26 11:25 myCryptodev -> ../dm-2
lrwxrwxrwx. 1 root root      7 Sep 26 09:50 rhel_itsoln1-root -> ../dm-0
lrwxrwxrwx. 1 root root      7 Sep 26 09:50 rhel_itsoln1-swap -> ../dm-1
[root@itsoln1 iso]#
```

---

By default, this command creates a symbolic link to /dev/dm-0. Upon verifying the symbolic link, you can see that the permissions are restrictive. In this case, only the root and members of the root group would have read/write access to the encrypted block device as shown in Example 3-10.

*Example 3-10 Permission restriction*

```
[root@itsoln1 iso]# ls -al /dev/dm-2
brw-rw----. 1 root disk 253, 2 Sep 26 11:25 /dev/dm-2
[root@itsoln1 iso]#
```

---

## Creating Linux file system and verifying the status

Now that the partition and the mapping to the partition are in place, look at the status of the LUKS mapper partition. As shown in Example 3-11, the partition is active and in use.

*Example 3-11 Device status*

```
[root@itsoln1 ~]# cryptsetup status myCryptodev
/dev/mapper/myCryptodev is active and is in use.
type:      LUKS1
cipher:    aes-xts-plain64:sha512
keysize:   512 bits
device:    /dev/dasdb1
offset:    4096 sectors
size:      43267712 sectors
mode:      read/write
[root@itsoln1 ~]#
```

---

Now, format the encrypted device partition using the EXT4 file system (Example 3-12). This file system formatting provides the ability to use the device partition just like any other partition.

*Example 3-12 Creating the file system*

```
[root@itsoln1 ~]# mkfs.ext4 /dev/dm-2
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
:
:
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000

Allocating group tables: done
Writing inode tables: done
```

```
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
[root@itsoln1 ~]#
```

---

After successful creation of the file system, you can mount the partition onto a folder as shown in Example 3-13.

*Example 3-13 Mounting the partition to a folder*

---

```
[root@itsoln1 ~]# mount /dev/dm-2 /mnt/mysecure_dev
```

---

### 3.3.3 Test for demonstrating encryption performance with hardware optimization

For the purpose of demonstration, we performed the following steps:

1. Mounted two disk partitions, one with hardware assisted encryption and the other without it.
2. Used the command line to create a large file with random numbers.
3. Ran the command line on both the disk partitions and recorded the elapsed time.

In the previous sections, we have already created and mounted a hardware assisted encrypted partition and also a plain disk partition as shown in Example 3-14.

*Example 3-14 Partition mount points*

---

```
[root@itsoln1 mnt]# ls -la
total 8
drwxr-xr-x.  4 root root   48 Sep 26 13:08 .
dr-xr-xr-x. 19 root root  247 Sep 22 09:38 ..
drwxr-xr-x.  3 root root 4096 Sep 26 17:00 mysecure_dev
drwxr-xr-x.  3 root root 4096 Sep 26 17:00 myunsecure_dev

[root@itsoln1 mnt]# mount
:
/dev/mapper/myCryptodev on /mnt/mysecure_dev type ext4
(rw,relatime,seclabel,data=ordered)

/dev/dasdc1 on /mnt/myunsecure_dev type ext4 (rw,relatime,seclabel,data=ordered)
```

---

#### Workload generation on the disk partition

For simplicity, we used the **dd** command to create a text file by populating random numbers using **/dev/urandom** devices. By using **/dev/urandom** for the generating random numbers, we created extensive resource consuming write operations (Example 3-15).

*Example 3-15 Executing workload on the encrypted partitions*

---

```
[root@itsoln1 ~]# time sh -c "dd if=/dev/urandom
of=/mnt/mysecure_dev/mycryptfile.txt bs=9048576 count=100"
100+0 records in
100+0 records out
904857600 bytes (905 MB) copied, 2.86093 s, 316 MB/s
```



```
real    0m2.886s
user    0m0.001s
sys     0m2.856s
```

---

Now we can proceed with running the same command line on the plain disk partition, mounted on /mnt/myunsecure\_dev (Example 3-16).

*Example 3-16 Running the workload on the encrypted partitions*

---

```
[root@itsoln1 ~]# time sh -c "dd if=/dev/urandom
of=/mnt/myunsecure_dev/mycryptfile.txt bs=9048576 count=100"
100+0 records in
100+0 records out
904857600 bytes (905 MB) copied, 7.44157 s, 122 MB/s
```

```
real    0m7.552s
user    0m0.001s
sys     0m2.946s
[root@itsoln1 ~]#
```

---

From this comparison, we can find that the hardware assisted encryption disk partitions have performed 3X times better than the normal or plain disk partition. We also witnessed a high data transfer rate with the encrypted partition.

In IBM Labs tests, with major workloads, the throughput to and from disk when using CPACF for encryption is much better than when the encryption is performed by software. There is only a relative small decrease when DES is used and a small decrease when AES is used for reading and writing to and from the encrypted file system if CPACF is used compared to no encryption.





## Pervasive encryption: Data in flight encryption

This chapter describes how to use the cryptographic functions of the IBM Z to encrypt data in flight. This technique means that the data is encrypted and decrypted before and after it is transmitted. We use OpenSSL to demonstrate the encryption of data in flight.

This chapter also shows how to customize the product to use the IBM Z hardware encryption features.

This chapter includes the following sections:

- ▶ Preparing to use OpenSSL
- ▶ Configuring OpenSSL
- ▶ Testing Hardware Crypto functions

## 4.1 Preparing to use OpenSSL

The Linux system we use for OpenSSL is already at a stage as described in Chapter 2, “Data security and Linux on IBM Z” on page 9. Therefore, the system is enabled to use the cryptographic hardware of the IBM Z. We also loaded the cryptographic device driver and the libica 3.0 package to use the crypto hardware.

To check whether OpenSSL is already installed in Red Hat Enterprise Linux 7.4, issue the command shown in Example 4-1.

*Example 4-1 Determining whether OpenSSL is installed*

---

```
[root@itsolnx2 /]# rpm -qa | grep openssl
openssl-1.0.2k-8.el7.s390x
openssl098e-0.9.8e-29.el7_2.3.s390x
openssl-libs-1.0.2k-8.el7.s390x
```

---

These packages are required for encryption, including hardware Cryptographic support:

- ▶ openssl
- ▶ openssl098e
- ▶ openssl-libs
- ▶ openssl-ibmca

During the installation of RHEL 7.4, the package openssl-ibmca was not automatically installed and needs to be installed manually as shown in Example 4-2.

*Example 4-2 Installation of openssl-ibmca package (not all lines of message shown)*

---

```
[root@itsolnx2 /]# yum install openssl-ibmca-1.3.0-2.el7.s390x.rpm
Installing:
  openssl-ibmca                               s390x
1.3.0-2.el7                                  /openssl-ibmca-1.3.0-2.el7.s390x
54 k
```

TDownloading packages:

```
Installing : openssl-ibmca-1.3.0-2.el7.s390x Installed:
  openssl-ibmca.s390x 0:1.3.0-2.el7
```

Complete!

---

Now all needed packages are successfully installed. At this moment only the default engine of OpenSSL is available as shown in Example 4-3.

*Example 4-3 Engine ibmca is not yet available for OpenSSL*

---

```
[root@itsolnx2 /]# openssl engine
(dynamic) Dynamic engine loading support
[root@itsolnx2 /]# openssl engine -c
(dynamic) Dynamic engine loading support
```

---

## 4.2 Configuring OpenSSL

To use the `ibmca` engine and to benefit from the Cryptographic hardware support, the configuration file of OpenSSL needs to be modified. To customize the OpenSSL configuration to enable dynamic engine loading for `ibmca`, complete the following steps:

1. Locate the OpenSSL configuration file, which in our Red Hat Enterprise Linux 7.4 distribution is in this subdirectory:

`/etc/pki/tls`

2. Make a backup copy of the configuration file as shown in Example 4-4.

*Example 4-4 Backup copy of openssl.cnf*

---

```
[root@itsolnx2 /]# ls -la /etc/pki/tls/openssl.cnf
-rw-r--r--. 1 root root 12376 Sep 25 14:25 /etc/pki/tls/openssl.cnf

[root@itsolnx2 /]# cp -p /etc/pki/tls/openssl.cnf
/etc/pki/tls/openssl.cnf.backup

[root@itsolnx2 /]# ls -al /etc/pki/tls/openssl.cnf*
-rw-r--r--. 1 root root 10923 Sep 25 14:25 /etc/pki/tls/openssl.cnf
-rw-r--r--. 1 root root 10923 Sep 25 14:26 /etc/pki/tls/openssl.cnf.backup
```

---

3. Some `ibmca` related content must be appended to the `openssl.cnf` file. This code comes with the `openssl-ibmca` package. Therefore, locate the `ibmca` package (Example 4-5) and look for a file called `openssl.cnf.sample.s390x`.

*Example 4-5 Locate the ibmca sample file*

---

```
[root@itsolnx2 /]# find / -name openssl.cnf.sample.s390x -type f
/usr/share/doc/openssl-ibmca-1.3.0/openssl.cnf.sample.s390x

[root@itsolnx2 /]# ls -al
/usr/share/doc/openssl-ibmca-1.3.0/openssl.cnf.sample.s390x

-rw-r--r--. 1 root root 1396 Mar 31 04:35
/usr/share/doc/openssl-ibmca-1.3.0/openssl.cnf.sample.s390x
```

---

4. Now, append the `ibmca`-related configuration lines to the OpenSSL configuration file (Example 4-6).

*Example 4-6 Append ibmca specific content to the OpenSSL configuration file*

---

```
[root@itsolnx2 /]# tee -a /etc/pki/tls/openssl.cnf <
/usr/share/doc/openssl-ibmca-1.3.0/openssl.cnf.sample.s390x
```

---

5. Make sure that the `ibmca` section was appended at the end of the OpenSSL configuration file as shown in Example 4-7.

*Example 4-7 Verify the append*

---

```
[root@itsolnx2 /]# grep -n ibmca_section /etc/pki/tls/openssl.cnf
371:ibmca = ibmca_section
374:[ibmca_section]
```

---

6. The reference to the `ibmca` section in the OpenSSL configuration file needs to be inserted. Therefore, insert the following line as shown in Example 4-8:

```
openssl_conf = openssl_def
```

*Example 4-8 Insert ibmca reference (not whole file shown here)*

---

```
[root@itsolnx2 ~]# cat /etc/pki/tls/openssl.cnf
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

# This definition stops the following lines choking if HOME isn't
# defined.
HOME               = .
RANDFILE           = $ENV::HOME/.rnd
openssl_conf = openssl_def #<== line inserted

# Extra OBJECT IDENTIFIER info:
#oid_file           = $ENV::HOME/.oid
oid_section        = new_oids

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
```

---

7. Check the value of the dynamic path variable and adjust it if needed (Example 4-9).

*Example 4-9 Check the dynamic path variable*

---

```
[root@itsolnx2 ~]# grep dynamic.path /etc/pki/tls/openssl.cnf
# Set the dynamic_path to where the libibmca.so engine
dynamic_path = /usr/lib64/openssl/engines/libibmca.so

[root@itsolnx2 ~]# ls -la /usr/lib64/openssl/engines/libibmca.so
-rwxr-xr-x. 1 root root 50072 Mar 31 04:35
/usr/lib64/openssl/engines/libibmca.so
```

---

You can find additional materials (such as the `openssl.cnf` file, which is customized for this paper) at the following website:

<ftp://www.redbooks.ibm.com/redbooks/REDP5464>

Example 4-10 shows the inserted `ibmca` section.

*Example 4-10 The ibmca section of the openssl.cnf file*

---

```
ibmca = ibmca_section
```

```
[ibmca_section]
```

```
# The openssl engine path for libibmca.so.
# Set the dynamic_path to where the libibmca.so engine
# resides on the system.
dynamic_path = /usr/lib64/openssl/engines/libibmca.so
engine_id = ibmca
init = 1
```

```
#
# The following ibmca algorithms will be enabled by these parameters
# to the default_algorithms line. Any combination of these is valid,
# with "ALL" denoting the same as all of them in a comma separated
# list.
#
# RSA
# - RSA encrypt, decrypt, sign and verify, key lengths 512-4096
#
# RAND
# - Hardware random number generation
#
# CIPHERS
# - DES-ECB, DES-CBC, DES-CFB, DES-OFB, DES-EDE3, DES-EDE3-CBC, DES-EDE3-CFB,
#   DES-EDE3-OFB, AES-128-ECB, AES-128-CBC, AES-128-CFB, AES-128-OFB,
#   AES-192-ECB, AES-192-CBC, AES-192-CFB, AES-192-OFB, AES-256-ECB,
#   AES-256-CBC, AES-256-CFB, AES-256-OFB symmetric crypto
#
# DIGESTS
# - SHA1, SHA256, SHA512 digests
#
default_algorithms = ALL
#default_algorithms = RAND,RSA,CIPHERS,DIGESTS
```

---

At the end of the `ibmca` section in the OpenSSL configuration file, you can choose the scope of the engine. You can either use the engine with its full capabilities (the default configuration), or you can include or exclude `RAND`, `RSA`, `DSA`, `DH`, `MACs`, or the symmetric ciphers.

You might want to exclude algorithms if there is no access to a Crypto Express feature in the Linux server. In this case, it is possible to use the RSA algorithm implemented inside of OpenSSL instead of the software fallback of `libica`. The appropriate configuration in the OpenSSL configuration file for that scenario is shown in Example 4-11.

*Example 4-11 The `ibmca` section in OpenSSL configuration file without access to `CEX6S`*

---

```
#default_algorithms = ALL
default_algorithms = RAND,CIPHERS
```

---

## 4.3 Testing Hardware Crypto functions

Now that the customization of OpenSSL is done, test whether you can use the IBM Z hardware cryptographic functions.

First, check whether the dynamic engine loading support is enabled by default and the engine `ibmca` is available and used in the installation as shown in Example 4-12.

*Example 4-12 Check if dynamic engine loading support is enabled and `ibmca` is used*

---

```
[root@itsolnx2 ~]# openssl engine
(dynamic) Dynamic engine loading support
(ibmca) Ibmca hardware engine support
```

---

Next, check what algorithms are supported. The Dynamic engine support for ibmca is enabled for ciphers available by using CPACF and CEX6S. See Example 4-13.

*Example 4-13 Supported algorithms*

---

```
[root@itsolnx2 /]# openssl engine -c
(dynamic) Dynamic engine loading support
(ibmca) Ibmca hardware engine support
[RSA, DSA, DH, RAND, DES-ECB, DES-CBC, DES-OFB, DES-CFB, DES-EDE3, DES-EDE3-CBC,
DES-EDE3-OFB, DES-EDE3-CFB, AES-128-ECB, AES-192-ECB, AES-256-ECB, AES-128-CBC,
AES-192-CBC, AES-256-CBC, AES-128-OFB, AES-192-OFB, AES-256-OFB, AES-128-CFB,
AES-192-CFB, AES-256-CFB, SHA1, SHA256, SHA512]
```

---

### 4.3.1 Crypto Express6S card support for OpenSSL

Check the number of executed requests in the cryptographic adapter. A change of this counter will be observed if RSA requests that use the cryptographic adapter are executed. First, observe the number of the counter before we execute the test shown in Example 4-14.

*Example 4-14 Display the request counter of the Crypto Express6S card*

---

```
[root@itsolnx2 /]# cat sys/devices/ap/card01/request_count
71
```

---

Next, perform crypto operations that use the cryptographic adapter, especially RSA, as shown in Example 4-15.

*Example 4-15 Testing RSA requests*

---

```
[root@itsolnx2 /]# openssl speed rsa2048 -elapsed
You have chosen to measure elapsed time instead of user CPU time.
Doing 2048 bit private rsa's for 10s: 4800 2048 bit private RSA's in 10.00s
Doing 2048 bit public rsa's for 10s: 6666 2048 bit public RSA's in 10.00s
OpenSSL 1.0.2k-fips 26 Jan 2017
built on: reproducible build, date unspecified
options:bn(64,64) md2(int) rc4(8x,char) des(idx,cisc,16,int) aes(partial)
idea(int) blowfish(idx)
compiler: gcc -I. -I.. -I../include -fPIC -DOPENSSL_PIC -DZLIB -DOPENSSL_THREADS
-D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DKRB5_MIT -m64 -DB_ENDIAN -Wall -O2 -g
-pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong
--param=ssp-buffer-size=4 -grecord-gcc-switches -m64 -march=z196 -mtune=zEC12
-Wa,--noexecstack -DPURIFY -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_GF2m -DRC4_ASM
-DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DAES_ASM -DAES_CTR_ASM -DAES_XTS_ASM
-DGHASH_ASM
          sign    verify    sign/s verify/s
rsa 2048 bits 0.002083s 0.001500s    480.0    666.6
```

---

After the test is completed, display the number of requests that were executed by the Crypto Express card. The number is much higher than before, which indicates that the Crypto Express feature has been used. See Example 4-16.

*Example 4-16 Display the request counter of the Crypto Express6S card*

---

```
[root@itsolnx2 /]# cat sys/devices/ap/card01/request_count
11551
```

---



Another method to see whether the Crypto Express feature was used is the **lszcrypt** command with the **-VV** or **-VVV** option, as shown in Example 4-17.

Example 4-17 Using lszcrypt to display the request counter of the Crypto Card

```
[root@itsolnx2 /]# lszcrypt -VV
card01: CEX5C      online  hwtype=11 depth=8 request_count=11551

[root@itsolnx2 /]# lszcrypt -VVV
card01: CEX5C      online  hwtype=11 depth=8 request_count=11551 pendingq_count=0
requestq_count=0 functions=0x10000000
```

You can also use the **icastats** command, as shown in Example 4-18.

Example 4-18 Running the icastats command

[root@itsolnx2 /]# icastats   head -n 4 && icastats   grep RSA						
function	hardware			software		
	ENC	CRYPT	DEC	ENC	CRYPT	DEC
RSA-ME		6668			0	
RSA-CRT		4927			0	

### 4.3.2 CPACF support for OpenSSL

This section shows some tests where OpenSSL uses the CPACF functionality on our z14. We reset the counter before all of these tests. Example 4-19 shows the test results when using 3DES encryption.

Example 4-19 Using the Triple DES (3DES) encryption

```
[root@itsolnx2 /]# icastats -r

[root@itsolnx2 /]# openssl speed -evp des-ede3-cbc 2>/dev/null | tail -n 3
The 'numbers' are in 1000s of bytes per second processed.
type           16 bytes    64 bytes   256 bytes  1024 bytes  8192 bytes
des-ede3-cbc   161890.70k  405335.27k  651198.50k  777228.97k  819437.57k
```

Example 4-20 shows the test results when using AES-128 encryption.

Example 4-20 Using the AES-128 encryption

```
[root@itsolnx2 /]# openssl speed -evp aes-128-cbc 2>/dev/null | tail -n 3
The 'numbers' are in 1000s of bytes per second processed.
type           16 bytes    64 bytes   256 bytes  1024 bytes  8192 bytes
aes-128-cbc    185463.67k  666155.02k 1892743.34k 3728567.98k 5035483.14k
```

Example 4-21 shows the test results when using AES-192 encryption.

Example 4-21 Using the AES-192 encryption

```
[root@itsolnx2 /]# openssl speed -evp aes-192-cbc 2>/dev/null | tail -n 3
The 'numbers' are in 1000s of bytes per second processed.
type           16 bytes    64 bytes   256 bytes  1024 bytes  8192 bytes
aes-192-cbc    176656.06k  624829.16k 1725169.32k 3244657.66k 4265056.58k
```

Example 4-22 shows the test results when using AES-256 encryption.

*Example 4-22 Using the AES-256 encryption*

```
[root@itsolnx2 /]# openssl speed -evp aes-256-cbc 2>/dev/null | tail -n 3
```

The 'numbers' are in 1000s of bytes per second processed.

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
aes-256-cbc	174327.62k	609381.24k	1623142.06k	2871059.46k	3685310.46k

Check the counter again. The results show that 3DES and AES are using the CPACF as shown in Example 4-23.

*Example 4-23 The icastats command*

```
[root@itsolnx2 /]# icastats
```

function	hardware			software		
	ENC	CRYPT	DEC	ENC	CRYPT	DEC
SHA-1		484			0	
SHA-224		4			0	
SHA-256		403			0	
SHA-384		4			0	
SHA-512		4			0	
GHASH		988			0	
P_RNG		0			0	
DRBG-SHA-512		676			0	
RSA-ME		4			0	
RSA-CRT		4			0	
DES ECB	0		0	0		0
DES CBC	0		0	0		0
DES OFB	0		0	0		0
DES CFB	0		0	0		0
DES CTR	0		0	0		0
DES CMAC	0		0	0		0
3DES ECB	8		8	0		0
3DES CBC	59099961		68	0		0
3DES OFB	8		8	0		0
3DES CFB	8		8	0		0
3DES CTR	4		4	0		0
3DES CMAC	56		20	0		0
AES ECB	196		12	0		0
AES CBC	283000465		72	0		0
AES OFB	12		12	0		0
AES CFB	24		24	0		0
AES CTR	624		12	0		0
AES CMAC	384		60	0		0
AES XTS	8		8	0		0

You can compare the number from your z14 with the numbers that were measured with a z13 in a [published paper](#) on the IBM Support website.

### 4.3.3 Testing the encryption using SSH

Now that customization of OpenSSL is now done and the cryptographic hardware is verified, perform a test using SSH. First, reset the counter and log on to the Red Hat Enterprise Linux system from a workstation. Then check the counter to see whether the cryptographic hardware was used, as shown in Example 4-24.

*Example 4-24 Test with SSH*

```
[root@itsolnx2 ~]# icastats -r
```

function	hardware			software		
	ENC	CRYPT	DEC	ENC	CRYPT	DEC
3DES ECB	0		0	0		0
3DES CBC	0		0	0		0
3DES OFB	0		0	0		0
3DES CFB	0		0	0		0
3DES CTR	0		0	0		0
3DES CMAC	0		0	0		0

```
[root@itsolnx2 ~]#
```

```
[itsoworkstation@oc5258314303 ~]$ ssh -c 3des-cbc root@9.76.61.222
```

```
root@9.76.61.222's password:
```

```
Last login: Wed Sep 27 10:47:01 2017 from 9.145.24.52
```

```
[root@itsolnx2 ~]#
```

function	hardware			software		
	ENC	CRYPT	DEC	ENC	CRYPT	DEC
3DES ECB	2		2	0		0
3DES CBC	22		39	0		0
3DES OFB	2		2	0		0
3DES CFB	2		2	0		0
3DES CTR	1		1	0		0
3DES CMAC	14		5	0		0

```
[root@itsolnx2 ~]#
```

The results show that the cryptographic hardware was used.





# IBM Secure Service Container

This chapter describes important features of the IBM Secure Service Container (SSC) appliance and its advantages.

This chapter includes the following sections:

- ▶ Introduction to IBM Secure Service Container
- ▶ IBM Secure Service Container internals
- ▶ Installing and managing the appliance
- ▶ Key features of Secure Service Container

## 5.1 Introduction to IBM Secure Service Container

The IBM Secure Service Container is a container technology through which you can more quickly and securely deploy firmware and software appliances on IBM Z and IBM LinuxONE™ (LinuxONE) servers.

A Secure Service Container partition is a specialized container for installing and running specific firmware or software appliances. An appliance is an integration of operating system, middleware, and software components that work autonomously and provide core services and infrastructures that focus on consumability and security.

## 5.2 IBM Secure Service Container internals

A Secure Service Container partition is a specialized container for installing and running specific firmware or software appliances. Secure Service Container partition contains its own embedded operating system, security mechanisms, and other features that are specifically designed for simplifying the installation of appliances, and for securely hosting them. Figure 5-1 provides a high-level view of the Secure Service Container components for hosting appliances on LinuxONE.

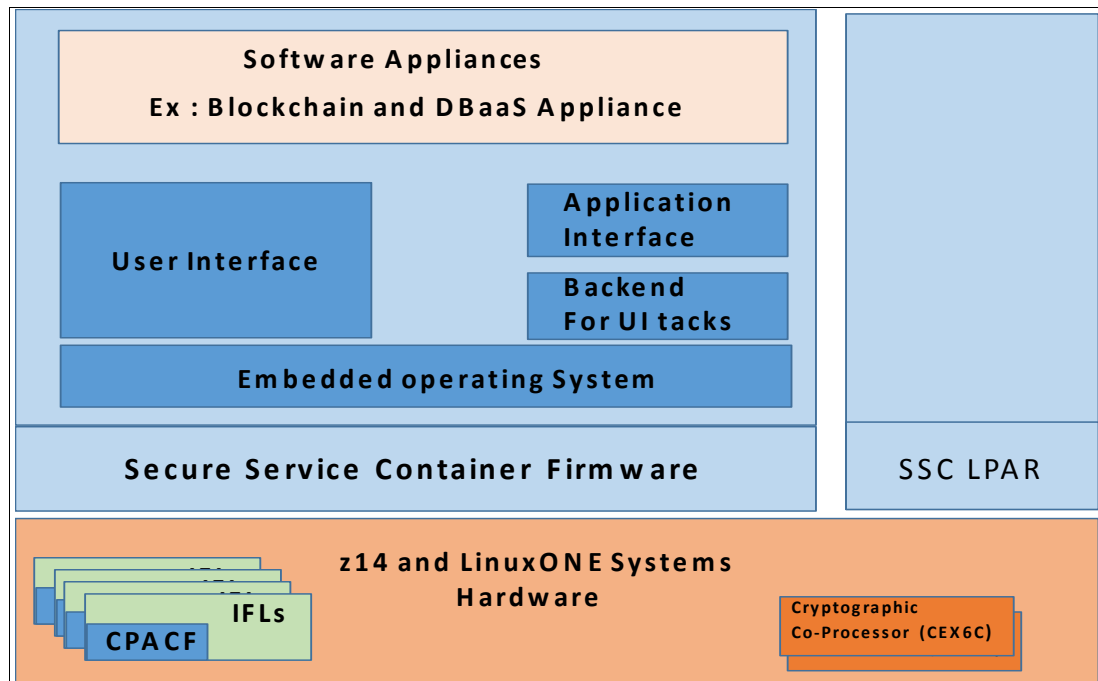


Figure 5-1 Secure service container internals

An appliance is an integration of operating system, middleware, and software components that work autonomously and provide core services and infrastructures that focus on consumability and security.

### 5.2.1 Secure Service Container boot environment

The design principle of Secure Service Container provides a fully tamper proof end-to-end secure environment (Figure 5-2). Harnessing the environment starts with a well-defined scope, encrypting container firmware and the container bootloader.

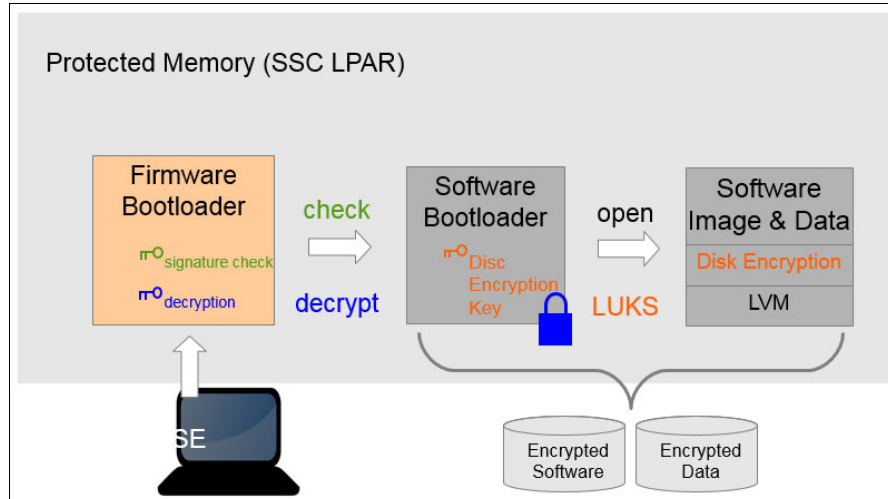


Figure 5-2 Secure Service Container boot sequence.

### 5.2.2 Boot sequence

The entire booting process is signed and encrypted so that they can only be executed within the protected and tamper-proof LPAR environment. The following is the boot sequence of a secure container service:

1. Firmware bootloader is loaded in memory.
2. Firmware loads the software bootloader from disk.
3. Check integrity of software bootloader.
4. Decrypt software bootloader.
5. Software bootloader activates encrypted disks.
6. Key stored in software bootloader (encrypted).
7. Encryption/decryption done in flight when accessing appliance code and data.

## 5.3 Installing and managing the appliance

This section covers the installation and configuration of the appliance, and subsequent management.

### 5.3.1 Installation and configuration

Secure Service containers provide a simplified mechanism for fast deployment and management of appliance-based solutions, including your O/S, applications, and services packaged as single secure solution.

For deploying secure service container, select **SSC** as the partition type during the Logical partition definition. From the HMC, Administrators would choose the mode of partition to be SSC, as shown in Figure 5-3.

The screenshot shows the 'Customize Image Profiles: PELCP02:APK1 : APK1 : General' window. On the left, a tree view shows 'PELCP02:APK1' expanded, with 'General' selected. The main area has the following fields:

- Profile name: APK1
- Description: This is the APK1 Image profile.
- Partition identifier: 33
- Mode: A list box showing options: ESA/390, ESA/390 TPF, Coupling facility, LINUX only, z/VM, and SSC (which is selected).
- Clock Type Assignment: Two radio buttons, 'Standard time of day' (selected) and 'Logical partition time offset'.
- A checkbox labeled 'Ensure that the image profile data conforms to the current maximum LICCC configuration.' which is checked.

Figure 5-3 HMC Partition configuration

**Note:** For more information about Secure Service container installation and configuration, see the [Secure Service Container User's Guide](#).

### 5.3.2 Managing the appliance

The appliances are pre-configured, up-dated, and serviced by using remote interfaces so that no root access to the underlying operating system is needed or permitted, even to rogue insiders.

Appliances that are designed to run in a Secure Service Container partition can use one or more Secure Service Container user interface (UI) widgets. Using them, you can view the Secure Service Container partition network and logon settings, request a dump of partition data for reporting a problem to IBM, and complete additional management tasks.

To access the installed appliance and view the Secure Service Container UI widgets, you need to know this information:

- ▶ The IP address for the Secure Service Container partition. Use the IP address of the network adapter that is specified in the image profile (standard mode system) or the partition definition (DPM-enabled system) for the Secure Service Container partition.
- ▶ The master user ID and password for the Secure Service Container partition. These values are specified in the image profile (standard mode system) or the partition definition (DPM-enabled system) for the Secure Service Container partition.

Communication with the appliance can be through the browser-based user interface or with REST APIs without any configuration.



## 5.4 Key features of Secure Service Container

IBM Secure Service Container provides the following optimized security functions for blockchain services:

- ▶ Protection from unauthorized access

Appliance code cannot be accessed even by platform or system administrators. Data access is controlled by the appliance, so unauthorized access is disabled. This protection is supported by a combination of signing and encrypting all data in flight and at rest. All access to memory is also removed. Firmware supports this protection with a secure boot architecture.

System administrators have the following limitations when blockchain is secured by IBM Secure Service Container:

- Cannot access nodes
- Cannot view the blockchain network

- ▶ Tamper protection

IBM Secure Service Container disables all external interfaces that provide LPAR memory access. An image boot loader is signed to ensure that it cannot be tampered or exchanged with a different one.

- ▶ Encrypted appliance disks

All code and data stored on disk is always encrypted by using the Linux encryption layer:

- Encapsulated operating system
- Protected IP
- Embedded monitoring and self-healing



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks

The following IBM Redbooks publication provides additional information about the topic in this document. Note that this publication might be available in softcopy only.

- *Security for Linux on System z*, SG24-7728

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Online resources

These websites are also relevant as further information sources:

- Secure Service Container User Guide

<http://www.ibm.com/support/docview.wss?uid=isg2bb79df265313634d85258088005188e3&aid=1>

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)







REDP-5464-00

ISBN 0738456586

Printed in U.S.A.

Get connected

